März/April 2017 | Nr. 554
www.elektormagazine.de

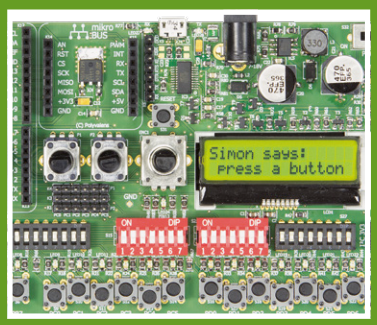
elektor

LEARN DESIGN SHARE

RGBDigit Clock

7-Segment-Displays in Farbe

OBD2-Handheld
Mit Raspberry Pi
und neuer Software



AVR-Entwicklungsboard
Summer, Taster, LEDs,
Potis, Drehgeber, Display

Bleiakku-Aktivator
Akku-Zustand
immer im Blick

In dieser Ausgabe:
8 Labor-Projekte
7 Homelab-Projekte
3x Grundlagen
und vieles mehr!

Starterkits und Entwicklungsboards • Android auf dem Raspberry Pi • **Rauschen und Empfindlichkeit bei Empfängern** • Theremin im Eigenbau • **Python für Mikrocontroller** • **Hausautomation: Internet-Gateway und drahtlose Endknoten** • Dock für BBC micro:bit • **Professioneller Zufallszahlen-Generator** • **PWM-Motorsteuerung** • Aktiver differentieller Tastkopf • **RFID-Starterkit für Arduino Uno** • **NEC-Fernbedienungssignale erzeugen** • Q&A: Elektromotoren • **MSR-Tool Swiss Pi: Hardware-Erweiterungen** • Web-Recherche: Lautsprecherboxen berechnen • **Bemerkenswerte Bauteile: Magnetblasenspeicher**

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

4 198630 314904 04

Die Entwicklung integrierter grafischer Benutzeroberflächen war noch nie so einfach

Kostenlose grafische Entwicklungstools & Grafikbibliotheken für Ihre GUI-Anwendungen

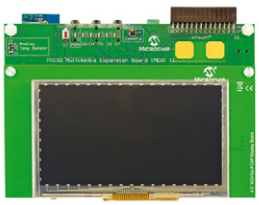
Microchip bietet für Ihre 32-Bit-Grafikanforderungen ein umfassendes Angebot an Entwicklungstools für grafische Benutzeroberflächen (GUIs), Software-Grafikbibliotheken und Hardware-Tools.

Unsere Grafiklösungen werden durch das kostenlose MPLAB® Harmony Software-Framework unterstützt und bieten Entwicklern die Wahl zwischen zwei Top-Tools:

1 Unser MPLAB® Harmony Graphics Composer arbeitet in Verbindung mit unserer MPLAB-Harmony-Grafikbibliothek, um Ihnen beim Erstellen professionell aussehender GUIs zu helfen, ohne eine einzige Zeile Code schreiben zu müssen!

ODER

2 Sie können SEGGER emWin Pro als Grafikbibliothek wählen und ein umfangreiches Angebot an Widgets sowie die SEGGER Toolchain nutzen.



Multimedia-Erweiterungsboard II (DM320005-2)

Machen Sie den ersten Schritt: Schulungsunterlagen, Dokumentation und Tools stehen zum Download bereit!

microchip DIRECT
www.microchipdirect.com

MPLAB HARMONY

MICROCHIP

www.microchip.com/MCU32GFX

Der Name Microchip, das Logo und MPLAB sind eingetragene Warenzeichen der Microchip Technology Incorporated in den USA und in anderen Ländern. Alle anderen Marken sind im Besitz der jeweiligen Eigentümer. ©2017 Microchip Technology Inc. Alle Rechte vorbehalten. DS60001462A. MEC2138Ger01/17

Impressum

48. Jahrgang, Nr. 554
März/April 2017

Erscheinungsweise: 12 x jährlich
(Elektor Doppelheft im Wechsel mit dem Elektor Business Magazin)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 02 41/955 09 190
Fax 02 41/955 09 013

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Hauptsitz des Verlags
Elektor International Media
Allee 1, NL-6141 AV Limbricht

Anzeigen:
Margriet Debeij (verantwortlich)
Tel. 02 41/955 09 174 / Fax 02 41/955 09 013
Mobil: +49 (170) 550 539 6
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/955 09 177 / Fax 02 41/955 09 013
Mobil: +31 6 510 530 39
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste ab 01.01.2017.

Distribution:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0
Fax 0 22 25/88 01-199

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für alle ausländischen Ausgaben inklusive Lizenzen. Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© 2017 elektor international media b.v.
Druck: Senefelder Misset, Doetinchem (NL)
ISSN 0932-5468

Mit Elektor auf Entdeckungsreise

Im letzten Editorial habe ich mich für Ihre rege Teilnahme an unserer Leserumfrage bedankt, und ich kann Ihnen versichern, dass wir Redakteure die Ergebnisse aufmerksam studiert haben. Zum Beispiel wissen wir jetzt, dass 86 % unserer Leser tatsächlich hin und wieder Projekte nachbauen, nur ein kleiner Teil greift gar nicht mehr zum Lötkolben. Es versteht sich daher von selbst, dass unsere Labor-Projekte das Herzstück von Elektor bleiben werden; ab jetzt kennzeichnen wir diese Artikel mit der Dachzeile „Labs Project“. Dazu gesellen sich die „Homelab Projects“ (bisher als Leser-Projekte bekannt). Diese Projekte werden von ihren Entwicklern immer besser ausgearbeitet, nicht zuletzt dank frei verfügbarer, professioneller Entwicklungstools. Nicht wenige der Autoren, die uns Projekte einsenden, bieten inzwischen sogar den Versand von Platinen an. Andere machen uns hochqualitative Vorlagen und CAD-Daten zugänglich, die wir zum – selbstverständlich kostenlosen – Download bereitstellen. Einige Beispiele solcher Projekte finden Sie in diesem Heft.
Auf den nachfolgenden Seiten sehen Sie, dass es keine strenge Einteilung unserer Artikel in die Abschnitte „Learn“, „Design“ und „Share“ mehr gibt. Projekte und Grundlagenartikel haben wir nun wieder bunt durchmischt. Die traditionelle Elektor-Philosophie, dass unsere Leser beim Durchblättern des Heftes auf eine Entdeckungsreise durch die Welt der Elektronik gehen, wird damit wieder deutlicher sichtbar. Im Inhaltsverzeichnis sind dagegen alle Projekte untereinander aufgeführt. Es zeigt sich damit auf den ersten Blick, wo der Fokus unserer Zeitschrift liegt.
Grundlagen, Vorstellungen neuer Entwicklungstools sowie Marktübersichten für Maker kommen dennoch nicht zu kurz. Besonders gefreut habe ich mich, dass wir rasch einen ersten Praxis-Artikel über Android auf dem Raspberry Pi verwirklichen konnten.
Viel Spaß beim Lesen!

Jens Nickel
Chefredakteur Elektor

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf
Leserservice:	Ralf Schmiedel
Korrekturen:	Malte Fischer
Internationale Redaktion:	Thijs Beckers, Jan Buiting, Mariline Thiebaut-Brodier
Elektor-Labor:	Harry Baggen, Ton Giesberts, Luc Lemmens, Denis Meyer, Jan Visser, Clemens Valens
Grafik & Layout:	Giel Dols

Inhalt

48. Jahrgang – Nr. 554
März/April 2017

Rubriken

3 **Impressum**

31 **Elektormagazine.de/Labs**
Projekte in der Pipeline

62 **HomeLab Helikopter**

74 **Projekt 2.0**
Korrekturen, Updates und Leserbriefe

82 **Retronik**
Made in Belgium

88 **Q&A**
Elektromotoren

96 **Bemerkenswerte Bauteile**
Magnetblasenspeicher

112 **Web-Recherche**
Lautsprecherboxen berechnen

130 **Hexadoku**

Hintergrund

12 **Starterkits und Entwicklungsboards**
Ein kleiner Überblick

42 **Rauschen und Empfindlichkeit bei Empfängern**
Eine Einführung

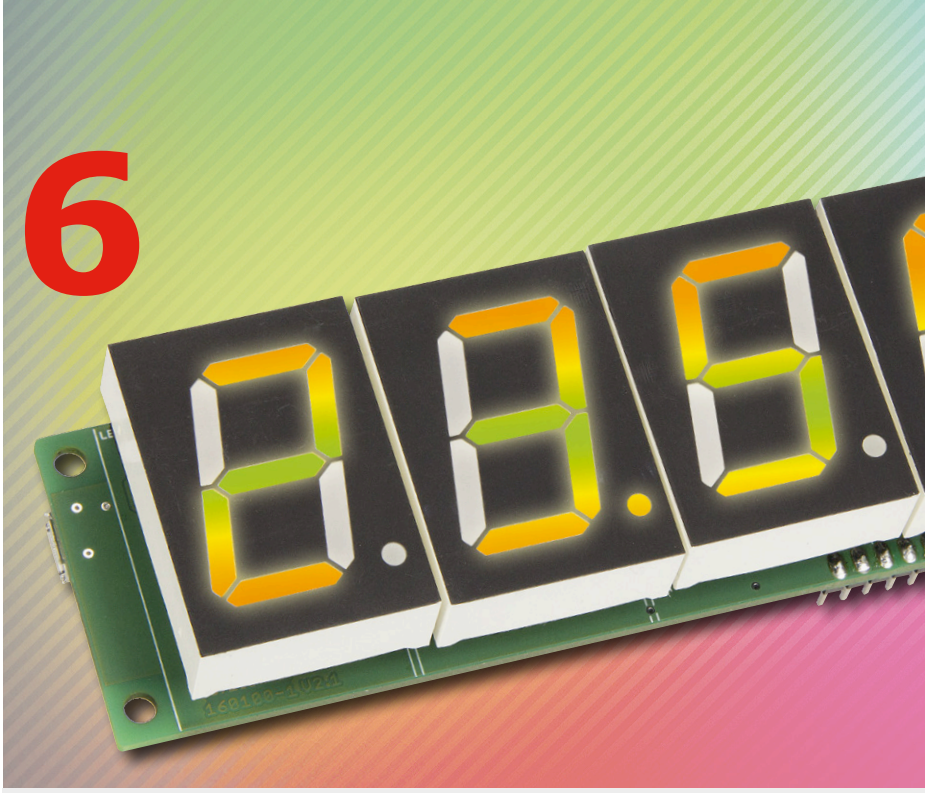
58 **Debugging und Dekodierung digitaler Kommunikation**
Mit dem SmartScope

64 **Review: STEMlab125-10**
Der kleine Bruder des Red Pitaya

76 **Android auf dem Raspi (1)**
Messen, Steuern, Regeln: GPIO-Pins

108 **RFID-Starterkit für Arduino Uno**
Temperaturmessung und Zugangskontrolle

118 **MicroPython**
Python für kleine Systeme



6

OBD2-Handheld mit dem Raspberry Pi

26

Mit neuer Diagnosesoftware

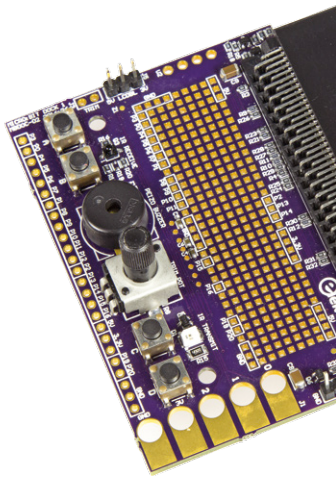
Das Modul Pi-OBD-HAT ist ein Add-on-Board für den Raspberry Pi, es verwandelt den Kleinstcomputer in ein OBD2-Diagnoseinterface. Ein Nachteil beim Handling war bisher, dass man die Lösung per Terminalprogramm ansteuern musste. Wesentlich einfacher und komfortabler geht es mit der hier vorgestellten OBD2-Diagnosesoftware HHGui.

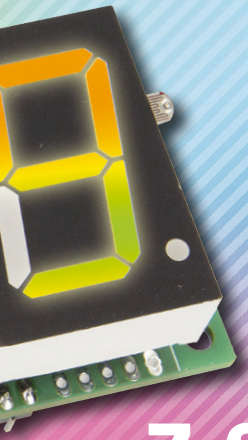
Projekte

6 **RGBDigit Clock**
Farbenfrohe 7-Segment-Displays zeigen Zeit und Klima

18 **IoT-Gateway und drahtlose Endknoten**
Teil 1: Die Hardware

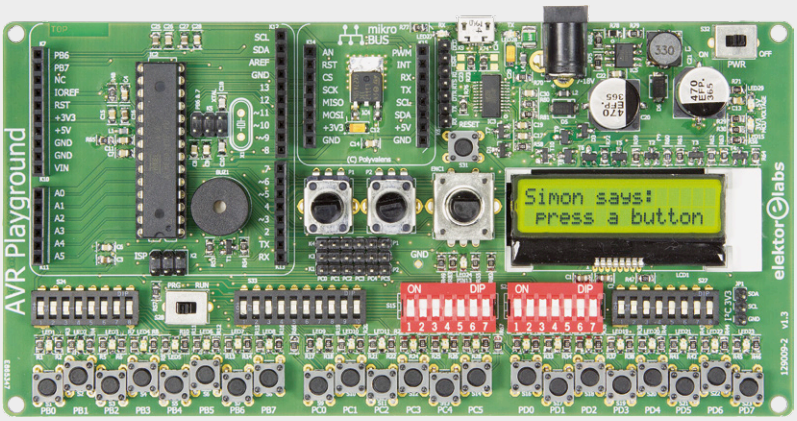
23 **NEC-Fernbedienungssignale**
Exklusiv erzeugt mit dem Elektor Uno R4





RGBDigit Clock

Farbenfrohe 7-Segment-Displays zeigen Zeit und Klima

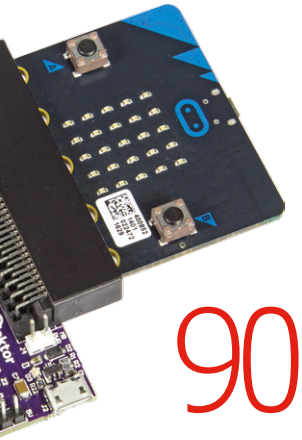


AVR Playground

Vereinfacht den Umgang mit Arduino

Das hier vorgestellte Board ist ein Hybrid aus Arduino Uno und einem konventionellen Entwicklungs-Board für Mikrocontroller. Es eignet sich besonders für die Arduino-Entwicklung – ohne schlechte Verbindungen, abgegangene Kabel und ähnliche Nervtötereien.

50



90

26 OBD2-Handheld mit dem Raspberry Pi
Mit neuer Diagnosesoftware

32 Swiss Pi – Flexibles Steuerungstool
Hardware-Erweiterungen

46 Theremin im Eigenbau
Mit JFETs statt Röhren

50 AVR Playground
Vereinfacht den Umgang mit Arduino

66 Zufallszahlen-Generator
Erfüllt jeden NIST-Test

90 Dock für BBC micro:bit

97 PWM-Motorsteuerung

100 Sensoren (4)
Für Arduino und Co.

106 Aktiver differentieller Tastkopf V2
Jetzt mit USB-Stromversorgung

114 Linux-Board Gnublin 2
Raspberry-Pi-Alternative zum Selbermachen

120 Bleiakku-Aktivator 0...30 V
Akku-Zustand im Blick

Vorschau

Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- Überblick: Starterkits für das IoT
- Li-Ion-Ladegerät
- Rollladen-Steuerung
- Network Connected Signal Analyzer II
- Mikrofon-Vorverstärker
- Außenbeleuchtung
- Supercaps
- 3D-Visualisierung mit Schrittmotoren

Und vieles mehr!

Änderungen vorbehalten.
Elektor Mai/Juni 2017 erscheint am 19. April 2017.
Verkaufsstellen findet man unter www.pressekaufen.de.

Vorschau Elektor Business

Mikrocontroller & Tools

- Von der Idee zur Serienreife mit 96 Boards
- Netzwerk-SoC
- Platinendrucker Voltera
- Warenwirtschaft für Start-ups

Und vieles mehr!

Änderungen vorbehalten.
Elektor Business 01/2017 erscheint am 1. März 2017.

RGBDigit Clock

Farbenfrohe 7-Segment-Displays zeigen Zeit und Klima

Wenn Sie „irgendetwas“ mit einem schicken Display bauen möchten, bauen Sie ... eine Uhr natürlich! Die RGBDigit Clock zeigt die Zeit auf tollen bunten 7-Segment-Displays der Marke RGBDigits an. Ein BME280-Breakout-Board ermittelt zusätzlich Temperatur, Feuchte und Luftdruck. Alle Daten werden nicht nur auf der fröhlichen Anzeige dargestellt, sondern auch von einem ESP12-Modul in das IoT übertragen.

Von Coen de Bruijn (Niederlande)

Die Uhr wird von dem ESP12-Modul gesteuert: Es synchronisiert die Uhr mit einem Internet-Zeitserver, ermöglicht die Einstellung der Uhr von jedem mobilen Gerät oder Computer im Netzwerk aus und überträgt Sensordaten via WLAN.

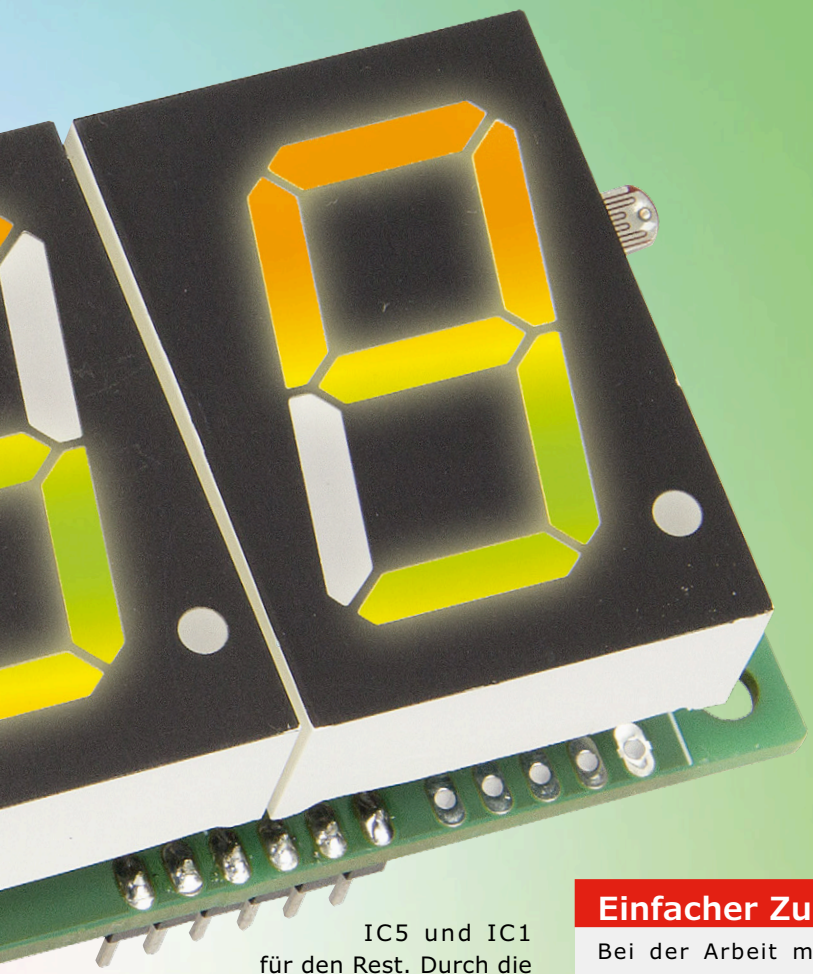
Was sind RGBDigits?

Die RGBDigits werden von einem niederländischen Unternehmen gleichen Namens hergestellt. Jedes Ziffern-Display besteht aus acht 5050-RGB-NeoPixel-LEDs (sieben Segmente plus Dezimalpunkt) mit integrierten Treiberchips, über die der Anwender Farbe und Helligkeit jedes Segments individuell über einen 3-Drahtbus (VCC, GND und DATA) steuern kann. Es können bis zu zehn Displays über ihre DATA IN- und DATA OUT-Pins aneinandergereiht werden.

Da sich jede Primärfarben-LED eines NeoPixels auf 256 Helligkeitsstufen einstellen lässt, sind 256 x 256 x 256 = 16.777.216 Farben möglich, also eine Farbtiefe von 24 Bit für jedes Segment!

Wie arbeitet die Schaltung?

Bild 1 zeigt die vollständige Schaltung dieses Projekts. Die Uhr wird über die Micro-USB-Buchse K1 mit 5 V versorgt. Die Stromversorgung wird von der 2-A-PTC-Sicherung F1 geschützt, wobei die Schottky-Diode D1 als Verpolschutz fungiert. Es gibt zwei 3,3-V-Spannungsregler, und zwar IC6 für die Qtouch-Touch-Sensoren IC4 und



INFOS ZUM PROJEKT

Testen & Messen
RGB-LED Uhr
Sensoren

Einsteiger
➡ Fortgeschrittener
Experte

Etwa 3 Stunden

Computer mit Arduino-IDE

Rund 100 Euro

IC5 und IC1 für den Rest. Durch die getrennte Versorgung werden Störungen sicher vermieden.

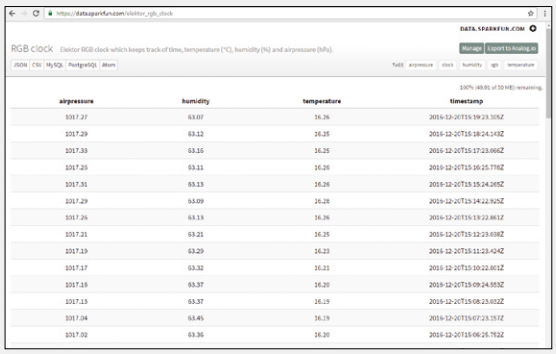
Mit den beiden Touchsensoren (Button0 und Button1) werden die Anzeigemodi der Uhr eingestellt. Es gibt auch zwei „echte“ Taster, S1 und S2, mit denen man die Uhr zurücksetzen und die Firmware flashen kann.

IC2 ist ein unidirektionaler Pegelumsetzer zwischen V+ (etwa 4,5 V, die Versorgungsspannung der Displays) und 3,3 V auf der ESP12E-Seite. Die Uhreneinstellungen werden beim Ausschalten immer in das EEPROM IC3 geschrieben. LDR R1 ermittelt die Umgebungshelligkeit, damit die Anzeige im Dunklen abgeblendet wird.

Ein FTDI-artiges 3,3-V-Kabel (oder ein anderer 3,3-V-USB/UART-Konverter) kann an K2 angeschlossen werden, um das ESP12E-Modul zu flashen oder auch um die Software zu debuggen.

K3 macht die 3,3-V-Stromversorgung und drei GPIO-Pins des ESP12E für eigene Entwicklungen und/oder zukünftige Erweiterungen zugänglich. An K4 kann ein BME280-BoB (Elektor-Shop 160109-2, **Bild 2**) zur Ermittlung von Klimadaten angeschlossen werden. Die Uhr läuft auch ohne das Breakout-Board, stellt dann aber

Einfacher Zugriff auf die Sensordaten



Bei der Arbeit mit diesem oder einem ähnlichen IoT-Projekt ist der Zugriff auf einen Datenspeicher-Server von Vorteil. Die Techies bei Sparkfun wissen das und haben einen solchen Service eingerichtet, der für jedermann frei ist. Über *data.sparkfun.com* kann man auf dem Server ganz bequem Daten speichern (und teilen!). Das Ganze funktioniert wie eine Datenbank, die Ihre (und andere) Daten in Form einer Tabelle (siehe **Screenshot**) präsentiert.

Zunächst erstellen Sie einen Datenstrom, indem Sie auf *create* klicken. Nach dem Ausfüllen der Felder erhalten Sie einen öffentlichen Schlüssel, einen privaten Schlüssel und einen Löschschlüssel. Diese Keys benötigt man zum Senden beziehungsweise Löschen von Daten an den beziehungsweise aus dem Stream. Unsere Uhr benötigt sowohl den öffentlichen als auch den privaten Schlüssel, um Daten an den Sparkfun-Datenserver zu senden.

Von jetzt an können Sie Ihre Daten mit einem Klick auf einen Knopf (rechts oben) zu *analog.io* exportieren, einem Online-Dienst für die Anzeige von (IoT-)Daten. Bei *analog.io* werden die nackten Daten/Messungen in eine schöne Grafik verwandelt.

natürlich keine Sensordaten bereit. Ein Onboard-BME280 war übrigens keine Option, da die Abwärme der Displays die Messdaten beeinträchtigen würde.

Sketch in den ESP12 flashen

Das WLAN-Modul ESP12E, das einen ESP8266 enthält, wird von der Arduino-IDE unterstützt. Um die Software in das Modul zu flashen, wählen Sie ESP8266 (NodeMCU) in der Arduino-IDE. Befolgen Sie die Anweisungen unter [1] zum Hinzufügen des ESP8266 sorgfältig, bevor Sie fortfahren.

Laden Sie das Software-Archiv für dieses Projekt (160100-11.zip) von unserer Website herunter [2], entpacken Sie es in einen Arduino-Sketchordner auf Ihrer Festplatte und achten Sie darauf, dass auch der Unterordner DATA (mit der ESP12-Webseite) darin enthalten ist.

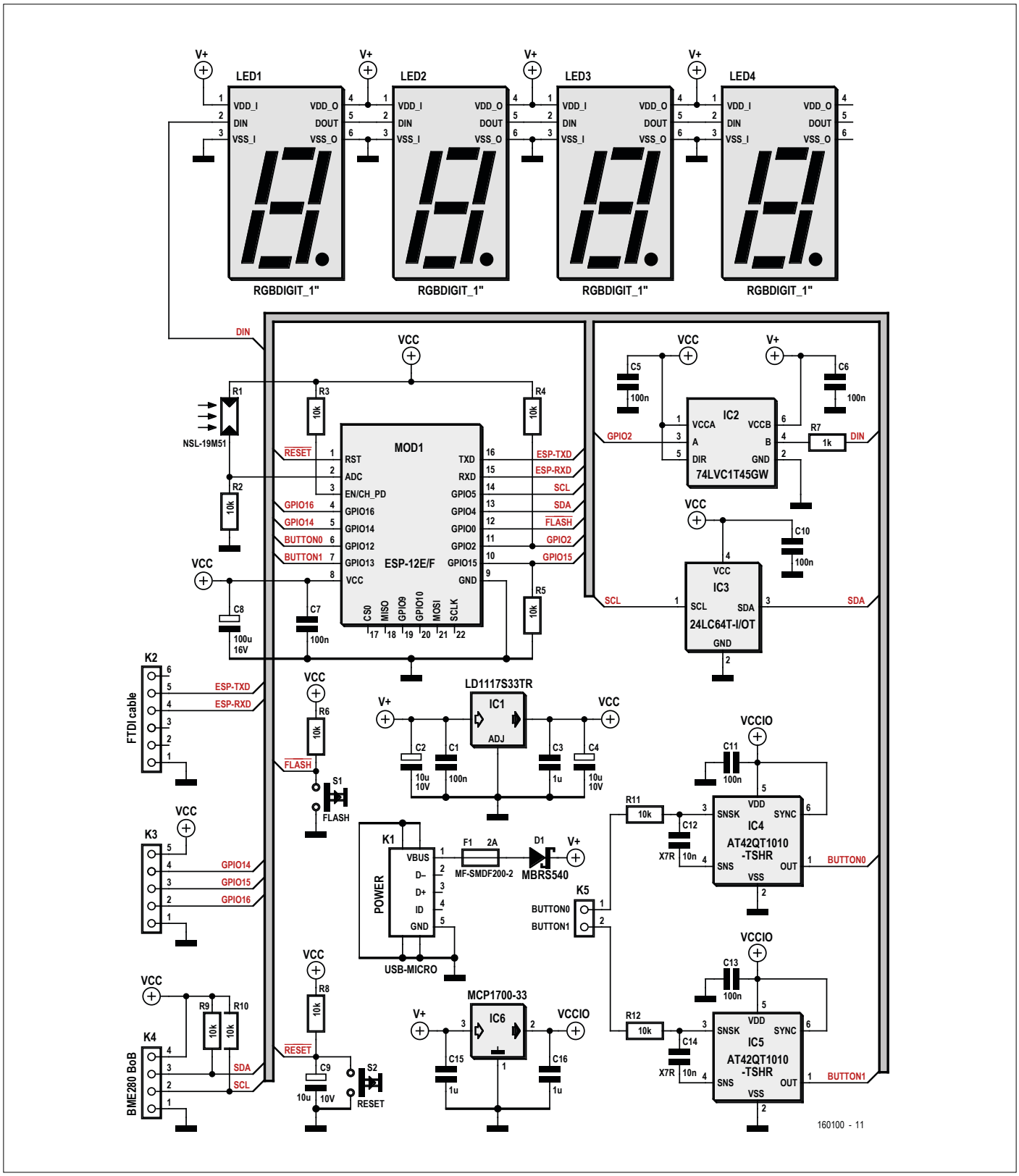


Bild 1. Viele Verbindungen: Um das Schaltbild übersichtlich zu halten, sind die Controllerleitungen als „Bus“ eingezeichnet.

Gehen Sie nun wie folgt vor:

- Legen Sie ein 3V3-FTDI-Kabel zwischen Computer und K2 der RGB-Uhr.
- Schließen Sie eine 5-V-Stromversorgung an die Micro-USB-Buchse K1 an.
- Öffnen Sie die Arduino-IDE.
- Wählen Sie im Menü Tools → Board das Board NodeMCU 1.0 (ESP12E-Modul).
- Wählen Sie im Menü Tools → Port den COM-Port, der dem FTDI-Kabel zugeordnet ist.
- Wählen Sie im Menü File → Open den Sketch „esp12_rgb_clock.INO“ aus dem Download-Paket von der Elektor-Webseite.
- Auf der RGB-Clock-Platine nun die Resettaste S2 drücken und halten, dann Flashtaste S1 drücken und halten, dann S2 freigeben und anschließend S1, um das Flashen der Firmware zu ermöglichen.
- Kompilieren und laden Sie den Sketch hoch (STRG + U).

Wenn der Compiler mit einem Fehler beendet wird, liegt dies wahrscheinlich an einer oder mehreren fehlenden Arduino-Bibliotheken. Der Compiler meldet den Namen der ersten fehlenden Bibliothek. Sie sollten daher vor dem Compilerstart überprüfen, ob alle Bibliotheken, die in den INCLUDE-Direktiven im Sketch verwendet werden, auch in Ihrer Arduino-IDE installiert sind, bevor Sie den Compiler starten.

Hochladen der Webseiten

Alle Einstellungen der Uhr können über WLAN konfiguriert werden. Die erste Verbindung sollte über den Access Point (AP) des ESP8266 erfolgen. Schließen Sie Ihren Computer (Smartphone/Tablet/Laptop/PC, wenn ein WLAN-Zugang besteht) an das offene Netzwerk mit dem Namen „RGB Clock“ an, das nach dem Flashen der Firmware in der Liste der verfügbaren WLAN-Netzwerke erscheint. Bevor man Zugriff auf die Einstellungen der Uhr bekommt, müssen erst die Webseiten der Uhr auf den ESP8266 hochgeladen werden. Unter [3] finden Sie im Abschnitt „Uploading files to the file system“ Anweisungen, wie Sie dies erledigen können. Hier in Kurzform:

- Laden Sie das Tool „Arduino ESP8266 filesystem uploader“ von [4] herunter.
- Entpacken Sie das Tool in das Tools-Verzeichnis. Der Pfad sollte etwa <home_dir>/Arduino/tools/ESP8266FS/tool/

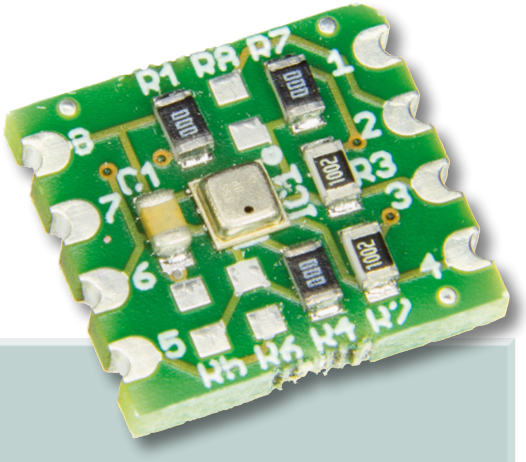


Bild 2. Unser BME280-Break-out-Board erfasst Temperatur, Luftdruck und Feuchte für die RGBDigit-Uhr.

STÜCKLISTE

Widerstände:

R1 = LDR NSL-19M51*
R2,R3,R4,R5,R6,R8,R9,R10,R11,R12 = 10 k
R7 = 1 k

Kondensatoren:

C1,C5,C6,C7,C10,C11,C13 = 100 n, 50 V, X7R, 0805
C2,C4,C9 = 10 µ, 10 V, Tantal, 1206
C3,C15,C16 = 1 µ, 50 V, X5R, 0805
C8 = 100 µ, 16 V, 2312
C12,C14 = 10 n, 50 V, X7R, 0805

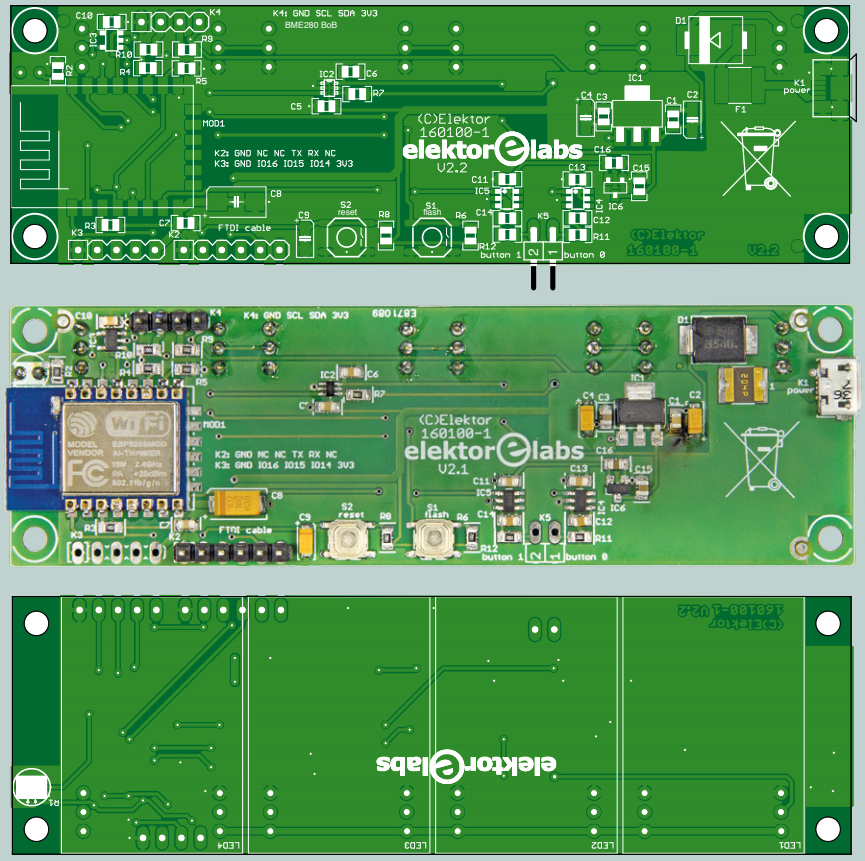
Halbleiter:

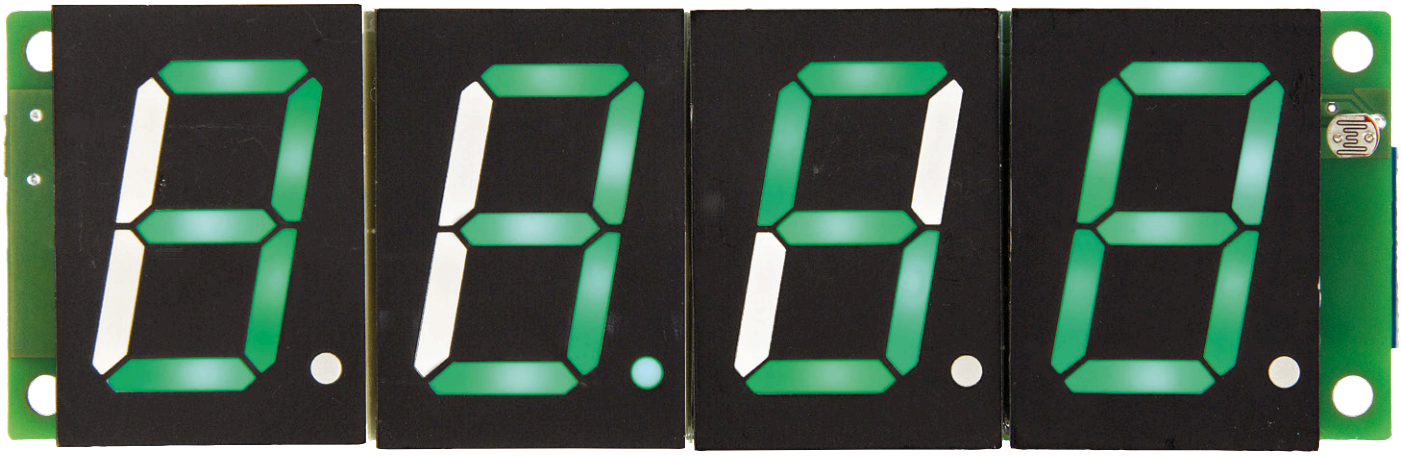
D1 = MBRS540
LED1,LED2,LED3,LED4 = 7-Segment-RGB-Display*
IC1 = LD1117S33TR
IC2 = 74LVC1T45GW
IC3 = I2C-EEPROM, 8K × 8 bit, Typ 24LC64T-I/OT
IC4,IC5 = Qtouch-Touchsensor, Typ AT42QT1010-TSHR
IC6 = MCP1700T-3302E/TT

Außerdem:

F1 = Rücksetzbare PTC-Sicherung, 2 A, Typ MF-SMDF200-2 (Bourns)
K1 = Micro-USB-Buchse, Typ B, Montage Unterseite
K2 = 1x6-polige Stiftleiste, Raster 0,1", gewinkelt
K3 = 1x5-polige Stiftleiste, Raster 0,1", gewinkelt
K4 = 1x6-polige Stiftleiste, Raster 0,1", gerade
K5 = 1x2-polige Stiftleiste, Raster 0,1", gewinkelt
MOD1 = ESP8266-12F
S1 = Taster
Platine 160100-1 v2.2

*wird auf der Platinenrückseite montiert





esp8266fs.jar lauten.

- Starten Sie die Arduino-IDE neu.
- Öffnen Sie den Sketch „esp12_rgb_clock.INO“ erneut, falls er nicht schon geöffnet ist.
- Überprüfen Sie, ob das richtige Board und der richtige COM-Port ausgewählt sind. Der Serial Monitor soll geschlossen sein.
- Resettaste S2 drücken und halten, dann Flashtaste S1 drücken und halten, dann S2 freigeben und anschließend S1, um das Flashen der Webseiten zu ermöglichen.
- Wählen Sie Tools → ESP8266 Sketch Data Upload. Die im Ordner DATA gespeicherten Daten werden nun in das ESP8266-Flash-Dateisystem hochgeladen. Abschließend zeigt die IDE die Mitteilung „SPIFFS Image Uploaded“ an.

Wenn Sie Lust haben, die Webseite der Uhr neu zu gestalten, tun Sie sich keinen Zwang an. Wie es geht, steht im Kasten „**Eine Webseite für den ESP12**“.

Der Spaß kann beginnen!

Verbinden Sie Ihren PC, Ihr Tablet oder Smartphone mit dem WLAN-Netzwerk „RGB Clock“ und öffnen Sie im Browser die Adresse 192.168.4.1:81. Wählen Sie links oben im Menü die Option „Wi-Fi settings“ und geben Sie die SSID und das Passwort Ihres WLAN-Netzwerks ein. Die Uhr benötigt dieses Netzwerk zur Zeitsynchronisation über einen Zeitserver (NTP). Im Sketch ist „time.nist.gov“ als Standard-NTP-Server eingetragen. Die Uhr sendet auch die Sensordaten an einen Server bei Sparkfun (siehe Kasten „**Einfacher Zugriff auf die Sensordaten**“).

Einige Bemerkungen:

Der Access Point des ESP12 ist auch weiterhin aktiv, nachdem Sie die WLAN-Einstellungen geändert haben. So können Sie diese oder andere Einstellungen jederzeit mit Ihrem Mobilgerät (im „RGB Clock“-Netzwerk) ändern.

Firewalls können – wie bei uns im Labor – das NTP-Protokoll eines Internet-Zeitservers blockieren, wodurch eine automatische Synchronisation der RGB-Uhr nicht möglich ist. Wir haben deshalb ein Smartphone als WLAN-Hotspot konfiguriert, um schnell die Zeitdaten für die Uhr abzurufen. Sobald sie synchronisiert ist, kann sie ohne NTP-Daten laufen. Alternativ können Sie die richtige Uhrzeit auch manuell auf der Webseite der Uhr einstellen.

Wenn Sie die Uhr an ein ungeschütztes Netzwerk, also ohne Kennwort anschließen möchten, aktivieren Sie das Häkchen bei „Unprotected Wi-Fi network“ in den WLAN-Einstellungen der Uhr.

Jedes Gerät, das auf Ihren WLAN-Router zugreifen kann, kann jetzt auch die Einstellungen der Uhr ändern. Geben Sie einfach „clock.local:81“ in die Adresszeile des Browsers ein, um auf die Webseiten der Uhr zuzugreifen. ⏮

(160100)

Eine Webseite für den ESP12

Die interne Website für den ESP12 wurde mit HTML-Kit292 (www.htmlkit.com/download) erstellt. Mit diesem freien HTML-Editor können Sie die Webseite entwerfen und publizieren. Unsere Webseite umfasst drei Dateien: *index.html*, *style.css* und *script.js*. Index.html ist die Hauptdatei und wird in HTML programmiert. Sie legt die Struktur der Website fest. In style.css wird definiert, wie die Seite aussieht. Und Script.js bestimmt, was die Seite macht.

Die ESP12-Webseite besteht aus mehreren -Elementen, deren Layout und Funktionalität in style.css und script.js definiert werden.

EINKAUFSZETTEL

- → **160100-1:** Leerplatine
- → **160100-91:** Aufgebautes Board
- → **160109-91:** Fertig aufgebautes BME280-BoB-Modul (optional)
- → **SKU 17789:** RGBDigit 7-Segment-Display

Weblinks

[1] www.arduinesp.com/getting-started

[2] www.elektormagazine.de/160100

[3] https://github.com/esp8266/Arduino/blob/master/doc/filesystem.md

[4] https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip

Die Elektor-Community

LEARN DESIGN SHARE

82	248153	1040	489	235332
Länder	Mitglieder	Experten & Autoren	Literatur	Monatliche Besucher

Elektor durchbricht die Schranken einer Zeitschrift und wird zur Community aktiver E-Ingenieure - vom Anfänger bis zum Profi – begierig, überraschende Elektronik zu lernen, zu entwickeln, zu teilen.

Elektor-Shop: 24 Stunden an 7 Tagen der Woche für jeden Elektroniker geöffnet! Dauerhafter Rabatt von 10% für alle GOLD- und GREEN-Mitglieder. **www.elektor.de**

Elektor-Zeitschrift: 12 Ausgaben (6 Doppelhefte + 6 Business-Ausgaben) pro Jahr voll gepackt mit Elektronik-Projekten, Artikeln, Besprechungen, Tipps und Tricks. **www.elektormagazine.de**

Elektor-Platinen-Service: Bestellung von Platinen als Einzelstück oder Kleinserie. **www.elektorpcbservice.de**

Elektor wöchentlich & papierlos: Wöchentlicher digitaler Newsletter. Kostenlos und aktuell. **www.elektor.de/newsletter**

Elektor Academy: Webinare, Seminare, Präsentationen, Workshops, DVDs und mehr = praxisorientiertes Lernen. **www.elektor.de**

Elektor-Bücher & DVDs: Arduino, Raspberry Pi, Mikrocontroller und vieles andere mehr. Im Online-Shop mit 10% Rabatt für Mitglieder! **www.elektor.de/bucher**

Elektor.TV: Reviews, Eindrücke, Unboxings und persönliche Journale. Anschauen heißt Erfahrung sammeln. **www.elektor.tv**

Elektor-Labs: Eigene Projekte vorstellen – von Anderen lernen – Anderen helfen und mit Anderen teilen. Elektor macht mit und testet Ihre Ideen! **www.elektormagazine.de/labs**

Treten Sie dem weltweit größten Elektroniker-Netzwerk bei!

GREEN 1,78 €/Woche

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 6x Elektor Business (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

Unsere Empfehlung

www.elektor.de/green-mitglied

GOLD 2,45 €/Woche

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 6x Elektor Business (Digital)
- ✔ 6x Elektor-Doppelheft (Print)
- ✔ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✔ Elektor Jahrgangs-DVD

www.elektor.de/gold-mitglied

GRATIS

- ✘ Zugang zum Elektor-Archiv
- ✘ 10% Rabatt auf Shop-Produkte
- ✘ 6x Elektor-Doppelheft (Digital)
- ✘ 6x Elektor Business (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

www.elektor.de/newsletter

facebook.com/ElektorDE

twitter.com/ElektorDE

Starterkits und Entwicklungsboards

Ein kleiner Überblick

Genau vor einem Jahr ging es an dieser Stelle um neue Boards und Mikrocontroller. In dieser Neuauflage unseres Überblicks (der natürlich wieder nicht vollständig sein kann) beschäftigen wir uns mit neuen Einsteiger-Boards und besonderen neuen Entwicklungsplattformen.

Von **Viacheslav Gromov** (D)
readers@gromov.de

Im letzten Jahr haben wir einen Blick auf die meist sehr günstigen Einsteiger-Boards und Starterkits geworfen [1]. Diese Einsteigerplattformen werden nicht so rasch an den technischen Fortschritt angepasst, da sie oft sehr universell sind und in großen Mengen produziert werden. Dies soll aber nicht heißen, dass die größeren Plattformen nicht zum Einstieg geeignet sind – ganz im Gegenteil. Oft ist es sogar besser, mit spezifischeren, komplexeren und dadurch vielleicht auch professionelleren MCU-Boards zu beginnen. Oder auch gleich ein Entwicklungssystem zu finden, das besonders auf die spätere Anwendung zugeschnitten ist. Die Einstiegshürde mag etwas höher sein, dafür ist aber auch der Lerneffekt größer. Außerdem ist es einfacher, später auf einen kleineren, weniger gut ausgestatteten Mikrocontroller umzusteigen als umgekehrt.

Deshalb führt dieser kleine Rundgang durch die MCU-Welt diesmal eher zu Exoten oder besonders neuen Boards, die alle sehr unterschiedlich und oft auf spezielle Anwendungsgebiete zugeschnitten sind. Dadurch schwanken sie auch im Preis von etwas über zehn bis zu mehreren hundert Euro. Die folgenden Abschnitte sind nach dem Preis geordnet. Es werden auch nicht die Standard-IDEs jeder MCU-Familie vorgestellt, sondern nur die Software-Neuerungen oder ganz neue (Zusatz-)Tools. Bestimmt ist auch etwas für Ihre nächste Entwicklung dabei!

Bevor es losgeht, möchten wir noch anmerken, dass die Flut der IoT-Boards mit Funk-Peripherie nachfolgend vernachlässigt wird. Diese Boards wollen wir in einem eigenen Artikel in der nächsten Ausgabe vorstellen.

Streichholzschachtelklein

Das bereits beim letzten Mal vorgestellte XMC-2Go-Board mit dem gut ausgestatteten XMC1100 (M0+, 48 MHz, 64 KB) als Basis wurde erweitert. Hinzu kamen drei Ausführungen mit SPI-Strom-Sensor TLI4970 (**Current Sensor 2Go**, circa 28 €, [2]), I²C-3D-Magnetfeld-Sensor TLV493D-A1B6 (**3D Magnetic Sensor 2Go**, circa 27 €, [2]) und SPI-H-Brücken-IC IFX9201SG (**H-Bridge Kit 2Go**, etwa 13 €, [3]) (**Bild 1**). Jeder der Sensoren hat seine besonderen Eigenschaften.

Der TLI4970 misst den intern durch einen Leiter fließenden Strom mit zwei integrierten Hall-Sensoren. Durch die verwendete Differenzmessung ist das Ergebnis unempfindlich gegen Umweltstörungen. Das Signal (bis zu ±50 A, 12 Bit, AC/DC) wird intern digitalisiert und mit einstellbaren Filtern aufbereitet beziehungsweise mit einem DSP weiter bis zur SPI-Schnittstelle verarbeitet, sodass sich das Ausgangssignal möglichst linear zum gemessenen Strom verhält.

Der TLV493D-A1B6 misst die Magnetfelder im Umfeld mit bis zu ±130 mT auf drei Achsen und löst sie jeweils mit 12 Bit auf. Zudem ist (auch zur Verbesserung der Messung) noch ein Temperatursensor an Board. Als Schnittstelle zum Mikrocontroller verwendet das Board I²C, wodurch ein sehr kleines 6-Pin-TSOP-Gehäuse möglich wird.

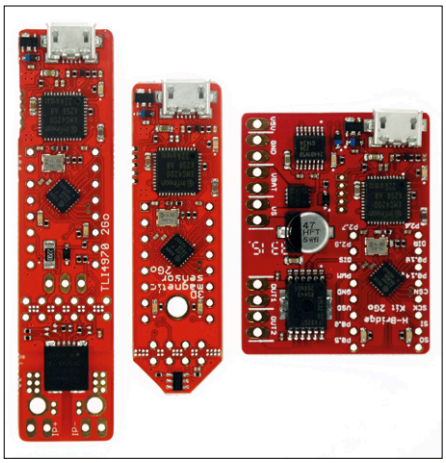


Bild 1. Current Sensor 2Go, 3D Magnetic Sensor 2Go und das H-Bridge Kit 2Go.

Der IFX9201SG ist unter den dreien das einzige IC, das keinen Sensor darstellt, sondern eine typische H-Brückenschaltung für kleine DC-Motoren oder andere induktive Lasten bis 6 A. Die Ansteuerung erfolgt größtenteils über den Direction-Pin

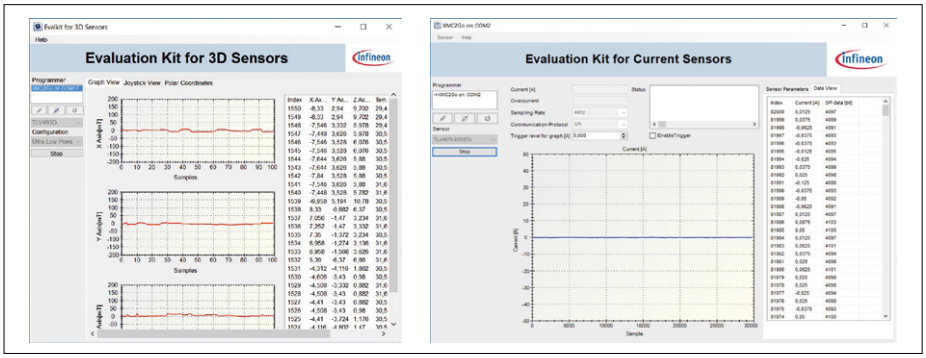


Bild 2. Für die beiden 2Go-Sensoren stehen kleine Software-Tools für Windows zur Verfügung, damit man sie schon vor der eigentlichen Programmierung ausprobieren kann.

und den PWM-Eingang (bis 20 kHz möglich!), die Überwachung kann durch die vorhandene SPI-Schnittstelle erfolgen.
Gemeinsam haben die Boards außer der XMC-Basis, dass die wichtigen MCU-Ausgänge herausgeführt sind und dass jeweils zwei User-LEDs vorhanden sind. Eine Besonderheit ist, dass man bei beiden Sensor-Boards die Sensoren von der Hauptplatine abtrennen kann. Beim H-Bridge-Kit ist das nicht möglich. Dafür gibt es zusätzliche Spannungsversorgungsanschlüsse, um das Platinchen leicht in eigenen Anwendungen verbauen zu können. Für den Magnetfeld-Sensor steht noch zusätzlich eine Joystick- oder Drehknopf-Erweiterung zur Verfügung, die dann aus dem Sensor ein stufenloses Bedienelement macht. Für jedes 2Go gibt es einige einfach zu nutzende Basis-Bibliotheken und –Projekte für die bekannte DAVE-Umgebung, außerdem für die Sensoren noch zusätzliche Software-Tools (**Bild 2**).

Mit Touch-Display für 25 €

STMicroelectronics war schon immer bekannt für die sehr günstigen, aber dennoch alles andere als spartanisch ausgestatteten Entwicklungsboards. Nun hat sich ST aber wohl selbst übertroffen. Das **STM32F412G Discovery-Kit** (**Bild 3**) [4], das auf einem neuen Vertreter der 32-Bit-ARM-Cortex-M4-STM32F4-Familie, dem STM32F412ZG (100 MHz, 1 MB) beruht, bietet folgende Highlights: ein 240x240-TFT-Touch-Display, ein Audio-Ausgang mit Codec-Chip und Klasse-D-Verstärker, zwei digitale MEMS-Mikrofone, ein 128 MBit-Quad-Zusatzspeicher, ein microSD-Slot und eine MicroUSB-Schnittstelle.
Natürlich sind auch zahlreiche MCU-GPIOs (**G**eneral **P**urpose **I**nput/**O**utput) herausgeführt, sogar in Form von Buchsenleisten im Arduino-Layout, was vor einiger Zeit fast nur bei Nucleo-Boards der Fall war. Die typische Grundausstattung eines Discovery-Boards ging dennoch nicht verloren: vier User-LEDs, ein Joystick (statt User-Tastern) und der gutmütige ST-Link/V2-1-Debugger. Schon wenn man das Board an die USB-Spannungsversorgung anschließt, erhält man allein wegen des Demoprogramms ein Gefühl für die unglaublich hohe Leistungsfähigkeit dieses unscheinbaren Mikrocontrollers – dort ist nämlich unter anderem ein Video- und Audio-Player mit Display, Audio-Out und Speicherkartenslot realisiert. Und all das bekommt man für gute zwanzig Euro – so viel, wie man etwa auch für einen Arduino Uno hinblättern muss.

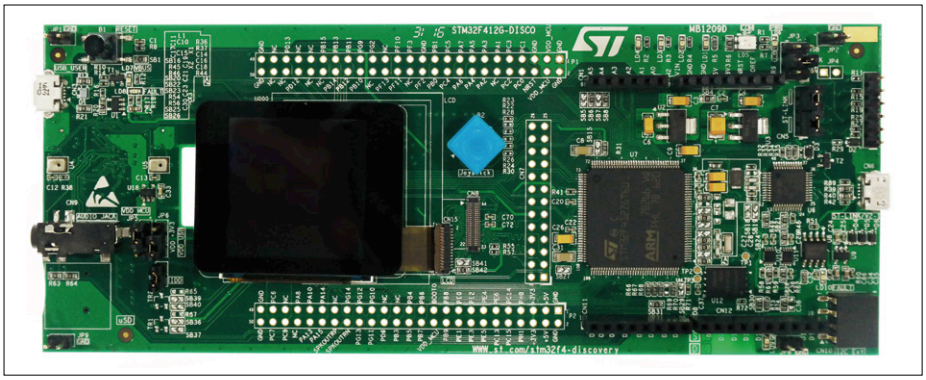


Bild 3. Durch das längliche Design des STM32F412G-DISCO kommen sich die Arduino- und die anderen Anschlüsse nicht ins Gehege, wenn ein Shield aufgesteckt ist.

Programmieren und debuggen kann man das Board am besten mit Keil (**Bild 4**), IAR oder openSTM32. Eine zentrale Rolle spielen der schon bekannte STM32CubeMX-Konfigurator sowie die STM32Cube-Software-Bibliotheken und -Beispielprogramme.

ATtiny schließt auf

Vor wenigen Jahren galten die ATtiny-MCUs als deutlich schwächer und schlechter ausgestattet als die großen Mega-Brüder. Zur Electronica 2016 hat man aber die ATtiny-Familie um 8xx-Typen mit 20 MHz, 14…24 Pins sowie 4 KB oder 8 KB Flashspeicher erweitert und dazu ein Xplained-Miniboard präsentiert. Spätestens jetzt können all die Megas und die ARMs nicht mehr verächtlich auf die Tiny-Mikrocontroller herabschauen, die jetzt zum großen Teil die gleichen internen Strukturen und Peripherie-Elemente der großen AVR- und sogar ARM-SAM-Vertreter aufweisen. Ein deutliches Anzeichen dafür ist das Event-System sowie die so genannten **C**ore **I**ndependent **P**eripherals (CIP), etwa der **P**eripheral **T**ouch **C**ontroller (PTC), der selbstständig und ohne Belastung der CPU arbeitet.
Seit neuestem gibt es aber auch ein **ATtiny817 Xplained Pro Board** (für etwa 36 €) mit dem gleichen Controller (**Bild 5**) [5]. Dies ist ein klares Zeichen für den Aufstieg der Tiny-Familie, auch, weil das Board beziehungsweise die MCU von der START-Onlineumgebung [6] unterstützt wird (**Bild 6**), in der man schon im Browser die wichtigsten Bibliotheken und Peripherie-Einstellungen in ein neues Projekt integrieren kann, bevor man zur IDE (Atmel Studio) greift. Das Board ist wie fast alle Xplained-Pros mit zwei Touch-Buttons, Stiftleisten, einem **E**mbedded **D**e**b**u**g**ger (EDBG) und sogar einer einstellbaren Messschaltung für den Stromverbrauch ausgestattet. Der größte Vorteil gegenüber dem Mini-Board ist aber die Möglichkeit, Xplained-Pro-Erweiterungen von Atmel

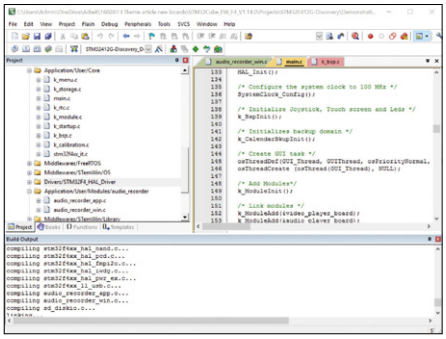


Bild 4. Wenn man die STM32-Cube-F4-Bibliotheken und Demoprogramme in der kostenlosen µVision-IDE von Keil nutzt, muss man darauf achten, dass man die Codebegrenzung nicht überschreitet.

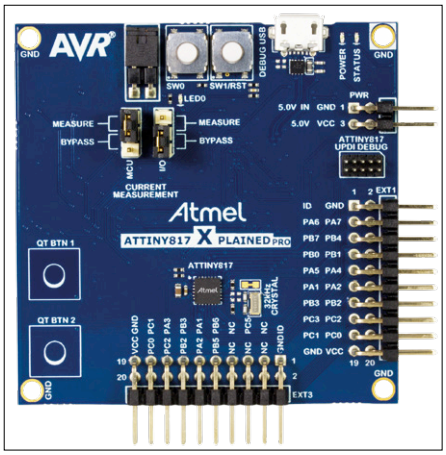


Bild 5. Das Attiny-817-Xplained-Pro-Board ist ungefähr so groß wie das Xplained-Mini, bietet aber mehr Möglichkeiten (Bildquelle: Microchip).

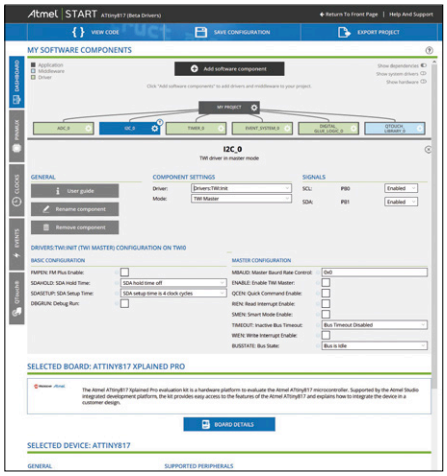


Bild 6. Ein Screenshot der START-Umgebung im Internet. Jedes der oben ausgewählten Elemente kann man noch genauer konfigurieren, bevor man das vorgefertigte Projekt in Atmel Studio 7 importiert.

zu benutzen – vom Temperatursensor bis zum OLED-Display ist so ziemlich alles dabei.

Kontakt zur analogen Welt

Als brandneues Mitglied der 32-Bit-ARM-Cortex-M0+-PSoC-Familie ist der PSoC-Analog-Coprocessor (48 MHz, 32 KB Flash, mit DMA) speziell auf die analoge Welt ausgerichtet. So stellt er (etwa in einem größeren Projekt) die ideale Schnittstelle zwischen dem analogen Bereich mit Sensoren und dem digitalen Bereich mit dem Hauptprozessor (in Datenblättern auch Host-Processor genannt) dar. Der Coprocessor verfügt über umfangreiche analoge Peripherie: zwei ADCs (12 Bit und 10 Bit), zwei 7-Bit-IDACs (zum Erzeugen von Strömen, meist für die Touch-Technologie), vier Operationsverstärker, zwei Komparatoren, CapSense Touch und ein **U**niversal **A**nalog **B**lock (UAB), den man als Filter, 14-Bit-ADC oder 13-Bit-DAC einsetzen kann. Zusammen mit einigen analogen Multiplexern hat man viele Möglichkeiten, ganz nach Wunsch ein analoges Peripherieelement mit einem Anschlusspin zu verbinden. Trotz des analogen Schwerpunkts gibt es genügend digitale Peripherie wie zum Beispiel Timer oder Schnittstellen. Besonders erwähnenswert sind der integrierte LCD-Treiber mit vielen Möglichkeiten (Schlafmodi bis 2,5 µA) sowie die minimale MCU-Versorgungsspannung von nur 1,71 V [7].

Um diese ganzen Möglichkeiten der flexiblen PSoC-Technologie zu präsentieren, wurde das rund 50 € teure „**CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit**“ [8] entwickelt (**Bild 7**). Dieses kleine Board verfügt über fünf analoge Sensoren: einen PIR-Bewegungsmelder, einen Temperatur-, einen Feuchte- und einen Lichtsensor sowie eine direkt auf die Platine gelayoutete Empfangs-Spule für einen induktiven Abstandssensor. Induktiv liegt im Trend, da dies viele Vorteile gegenüber der kapazitiven Technologie bietet. Der Sensor ist zwar ausgelegt, um den Abstand zur mitgelieferten Metallmünze zu detektieren, grundsätzlich kann aber auf ähnliche Weise ein Touch-Sensor aufgebaut werden, der die (kaum merkliche) Biegung der Metallplatte bei Berührung registriert. Für die aufgezählten Sensoren gibt es auf dem Board oft noch verschiedene Schalt-Möglichkeiten mit Schaltern oder Jumpern. Viele Anschlüsse sind auf die Pin-Leisten (unten Stift, oben Buchse) herausgeführt, wodurch dieses Board sowohl als (Arduino-)Shield fungieren als auch zusätzlich aufgesteckte (Arduino-)Shields ansteuern kann. Es gibt auch eine Grundausstattung an digitaler Beschaltung – eine User-RGB-LED und ein User-Taster. Als Brücke zwischen der Mini-USB-Buchse und dem Hauptcontroller dient der KitProg2-Programmer und -Debugger, mit dem man das Board ohne Zubehör direkt vom PSoC-Creator aus programmieren und debuggen kann (**Bild 8**).

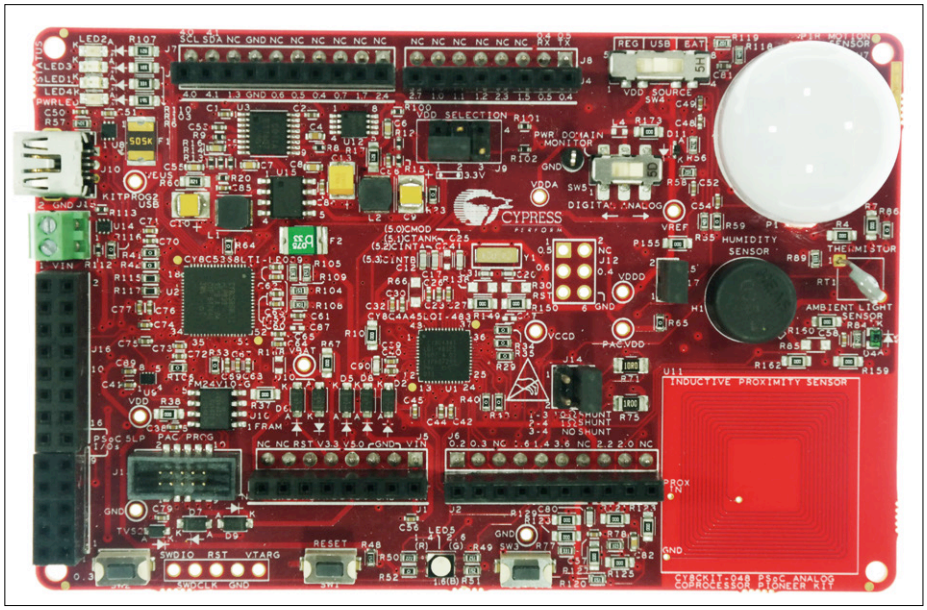


Bild 7. Der Inhalt des CY8CKIT-048: Außer Board und Kabel gibt es Buchsenleisten für Selbstbauerweiterungen sowie eine Metall-Münze zum Testen der induktiven Abstandserkennung.

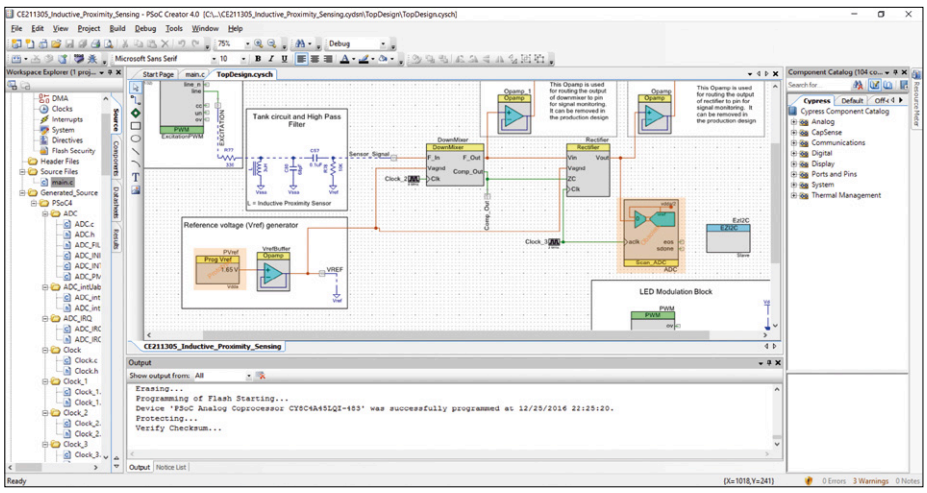


Bild 8. Hier ist die Innenschaltung der internen Peripherie zu sehen, die zur Auswertung des induktiven Sensors im PSoC Creator genutzt wird. Dadurch bleibt dem Entwickler ein Großteil der Codeerstellung erspart.

Gleich im großen Stil?

Letztes Jahr wurde von Renesas ein kleines, auf der 16-Bit-RL78-Familie basierendes Promo-Board vorgestellt. Nun preist der Hersteller aber schon länger die 32-Bit-ARM-Cortex-M-Synergy-MCU-Familie an. Normalerweise muss man für Starterkits von Renesas ganz schön tief in die Tasche greifen, beim **SK-S7G2-Starter Kit** mit Synergy-S7-MCU (240 MHz, 4 MB Code Flash, mit DMA) ist es jedoch anders (**Bild 9**) [9].

Das Kit wird als Einstiegsplattform in vielen Beispiel-Applikationen des Herstellers verwendet, für den relativ niedrigen Preis von rund 80 € verkauft und auf Messen sogar kostenlos verteilt. Der S7 (Cortex M4) hat seine Stärken im HMI-(**H**uman-**M**achine **I**nterface) und Connectivity-Bereich, vom Grafik-LCD-Controller über den JPEG-Codec bis zur Ethernet-Fähigkeit mit zwei integrierten MAC-Controllern reicht die Palette der Features. Gerade also in den Bereichen der Mensch-Maschine-Kommunikation und der Vernetzung, den Kernthemen von Industry 4.0, IoT und Smart Home, zeigt er sein Können. Dies spiegelt sich natürlich auch beim Board-Aufbau wider. Außer den unzähligen Stiftleisten mit MCU-Pins und Buchsenleisten im Arduino-Layout ist ein TFT-Touch-Display an Bord, mehrere Touch-Elemente, USB-, Audio- sowie Ethernet-Schnittstelle(n). Es gibt einige Erweiterungsmöglichkeiten, so kann man beispielsweise ein BLE-Zusatzmodul (**B**luetooth **L**ow **E**nergy) anschließen. Nicht zu vernachlässigen sind natürlich die drei User-LEDs und zwei User-Taster sowie natürlich der integrierte J-Link-On-Board-Debugger. Auf den teuren E1-Emulator kann bei diesem Kit also getrost verzichtet werden.

Ganz nebenbei: Es gibt zu diesem Kit und der gesamten Synergy-Familie ein sehr empfehlenswertes, brandneues Buch als Download [10].

Die Standard-IDE für die gesamte Synergy-Plattform ist zwar das bereits vom RL78 bekannte E²studio von Renesas, es gibt aber zusätzlich noch die Gallery (**Bild 10**), eine Plattform ausschließlich für Synergy [11]. Hier bekommt man immer die neuesten Bibliotheken, Software-Zusatztools (zum Beispiel zur Display-Grafikerstellung oder Sicherheits-Konfiguration) und natürlich viel Dokumentation und Support.

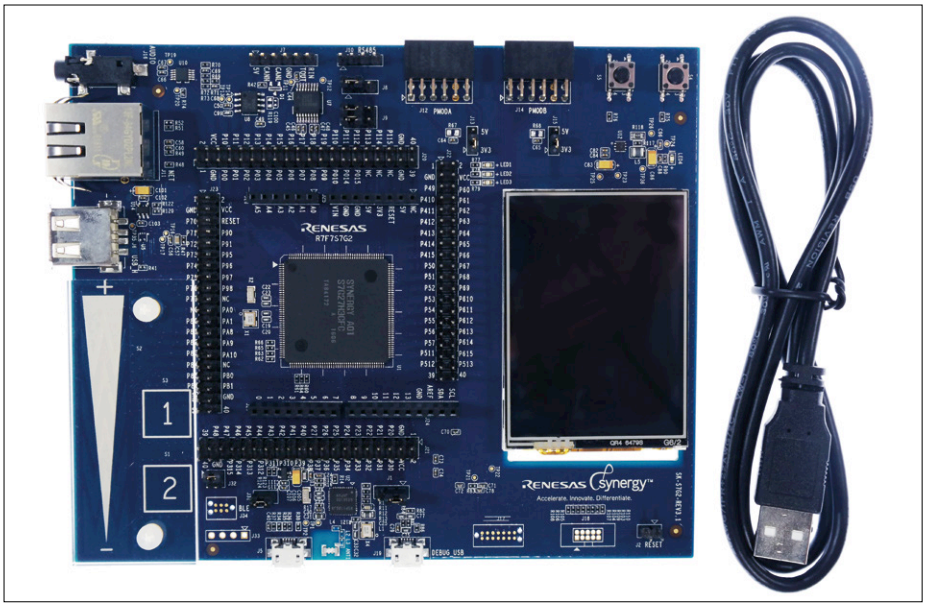


Bild 9. Das SK-S7G2 besteht nur aus Platine und USB-Kabel, ist aber sehr preiswert. Mehr wird zum Debuggen auch nicht benötigt.

Wireless-Board-Duo

Seit kurzem gibt es den KW41Z (48 MHz, 512 KB Flash, DMA) [12], einen 32-Bit-ARM-Cortex-M0+-Mikrocontroller von NXP, der sich völlig dem Thema Wireless widmet. Dabei kombiniert er grob gesagt die Fähigkeiten von KW21Z und KW31Z auf dem 2,4-GHz-Band, sodass er sowohl im Bluetooth- beziehungsweise BLE-Modus (**B**luetooth **L**ow **P**ower 4.2) als auch nach dem IEEE 802.15.4-Standard (dazu zählen dann auch weitere Protokolle wie etwa Thread) kommunizieren kann. Deshalb wird der KWZ41Z auch Dual-Mode-MCU genannt. Wie bei NXP (und früher Freescale) üblich, gibt es dazu auch ein Freedom-Kit namens

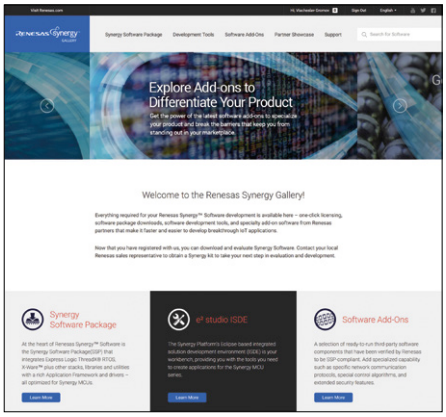


Bild 10. Unten auf dem Screenshot der Synergy Gallery sieht man die wichtigsten Inhaltsbereiche.

Bild 11. Das Board-Duo mit den beiden im Kit enthaltenen USB-Kabeln.

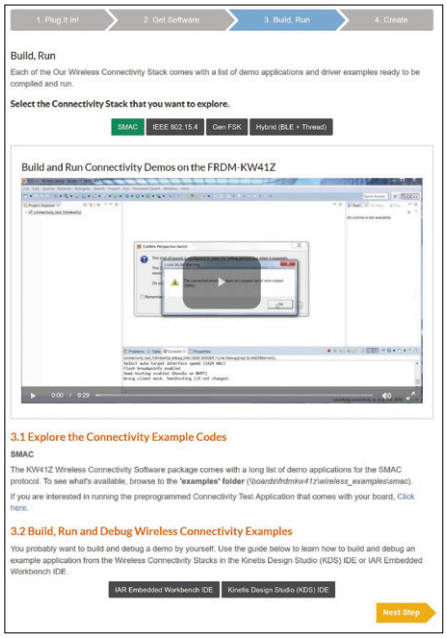


Bild 12. Nachdem alles installiert und eingerichtet ist, kann man mit den „Getting Started"-Videos die Beispielprogramme durchgehen.

FRDM-KW41Z (**Bild 11**), das praktischerweise aus zwei identischen Boards besteht, so dass man die Kommunikation direkt ausprobieren kann (rund 150 €). Die Boards besitzen außer dem OpenSDA-Debugger zum direkten Debuggen aus dem Kinetis Studio heraus eine auf der Platine gelayoutete Antenne und alternativ sogar eine Antennenbuchse. Außerdem gibt es noch eine Knopfzellenhalterung, eine User-RGB-LED, eine User-LED, zwei User-Taster, einen analogen Temperatursensor und den FXOS8700CQ-I²C-Sensor, der 3-Achsen-Beschleunigungs- und -Magnetfelddaten liefert.

Wie gewöhnlich liefert Freescale (nun NXP) eine einwandfreie Dokumentation. Überzeugend ist auch die interaktive „Getting Started"-Beschreibung auf der Produktseite (**Bild 12**) [13]. Dort wird Schritt für Schritt der Einstieg von der Installation der IDE und KW41Z-Software über das Ausprobieren der Beispielprogramme bis hin zum Erstellen des eigenen Projekts mit Hilfe von vielen Videos beschrieben.

Der digitale Analog-Experte

Neben all den berühmten Controller-Herstellern wie Microchip, ST, NXP & Co. vergisst man oft, dass zum Beispiel auch ein eher in der analogen Welt anzusiedelnder Hersteller wie Analog Devices MCUs produziert und sehr schöne Kits und Anwendungs-Beispiele anbietet. Eine der neuesten und bekanntesten MCUs aus der ARM-Cortex-M3-Serie ist der ADuCM3029 (26 MHz, 256 KB Flash). Die MCU ist - wie es sich für einen M3 gehört - in allen Bereichen solide ausgestattet [14]. Glänzen kann der Mikrocontroller mit seinen vier Stromsparmodi, bei denen der Verbrauch auf bis unter 60 nA abgesenkt werden kann.

Für 200 € bekommt man passend dazu ein **EVAL-ADuCM3029 EZ-KIT** (**Bild 13**), das im Wesentlichen aus einem MCU-Board mit zwei User-Tastern, drei User-LEDs, einem Summer, einem Beschleunigungs-/Temperatursensor ADXL363 bzw. ADT7420, einem (seriellen) Mini-USB-Anschluss, zwei Batterie-Halterungen, Arduino-Layout-Buchsenleisten und einem externen J-Link-LITE-Debugger besteht. Programmiert werden kann es mit den üblichen IDEs von Keil und IAR (dafür gibt es auch viel Software und Dokumentation) oder dem hauseigenen, aber leider nicht kostenlosen CrossCore-Embedded-Studio [15]. Ein nicht zu unterschätzender Gesichtspunkt, wieso man sein Projekt mit dieser MCU-Familie realisieren sollte, ist die Erfahrung des Herstellers in der analogen Welt. Analog Devices bietet auch Sensoren und andere Komponenten rund um die MCU an, für die entsprechende Bibliotheken und weiterer Support geboten werden. Empfehlenswert sind für die ersten Versuche die vielen erhältlichen (Sensor-)Shields.

Das Vordringen dieses Unternehmens in die digitale Welt ist auch am Beispiel des **ADZS-BF707-BLIP2-Boards** [16] und der damit zusammenhängenden Technologie auszumachen. Dieses rund 200 € teure Board verfügt über viele Schnittstellen und den mit 400 MHz getakteten 32-Bit-Mikrocontroller ADZS-BF707. Mit den beiden On-board-Kameras/Sensoren ist man in der Lage, Gegenstände, Menschen oder gar Gesichter zuverlässig zu erkennen. Für Entwicklungen mit dem Board benötigt man den zum Kit gehörenden ICE-1000-Emulator und auch das CrossCore-Embedded-Studio (eine Lizenz für 120 Tage ist im Lieferumfang enthalten). Es gibt aber auch ein schönes Tool, um nur einen ersten Eindruck zu erhalten.

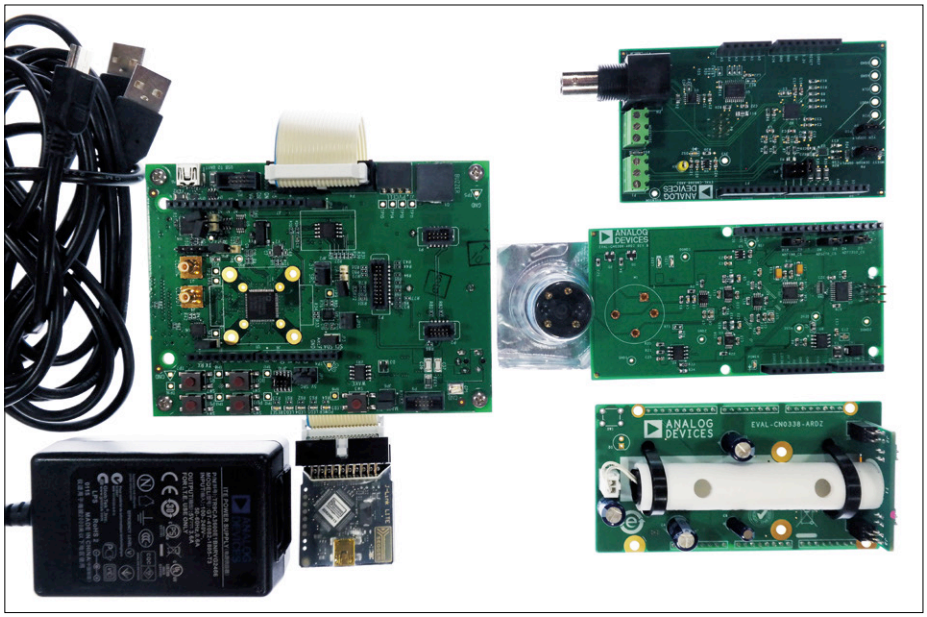


Bild 13. Die wichtigsten Bestandteile des Kits sind der Debugger, das Board, das USB-Kabel sowie das Stromversorgungs-Netzteil. Rechts daneben ist eine kleine Auswahl an A/D-Shields (hier: pH/Temperatur/Feuchtigkeit sowie Gassensoren) zu sehen.

Energy Harvesting in allen Facetten

Der Halbleiterhersteller Linear Technology hat jüngst zusammen mit Würth Elektronik, dem Experten in Sachen passiver Bauelemente und Steckverbindungen, das **Gleanergy Demonstration Kit** entwickelt, das rund 400 € kostet (**Bild 14**). Gleanergy ist ein Kunstwort, dass sich aus dem Englischen „to glean" für „sammeln" und „energy" zusammensetzt. Das Kit besteht hauptsächlich aus zwei Platinen und einem USB-Sender/Empfänger DC2274A-A [17].

Die erste Platine DC2344A [18] ist für den analogen Teil des Energy Harvesting verantwortlich. Auf ihr sind eine Batterie („primary battery"), ein Akku

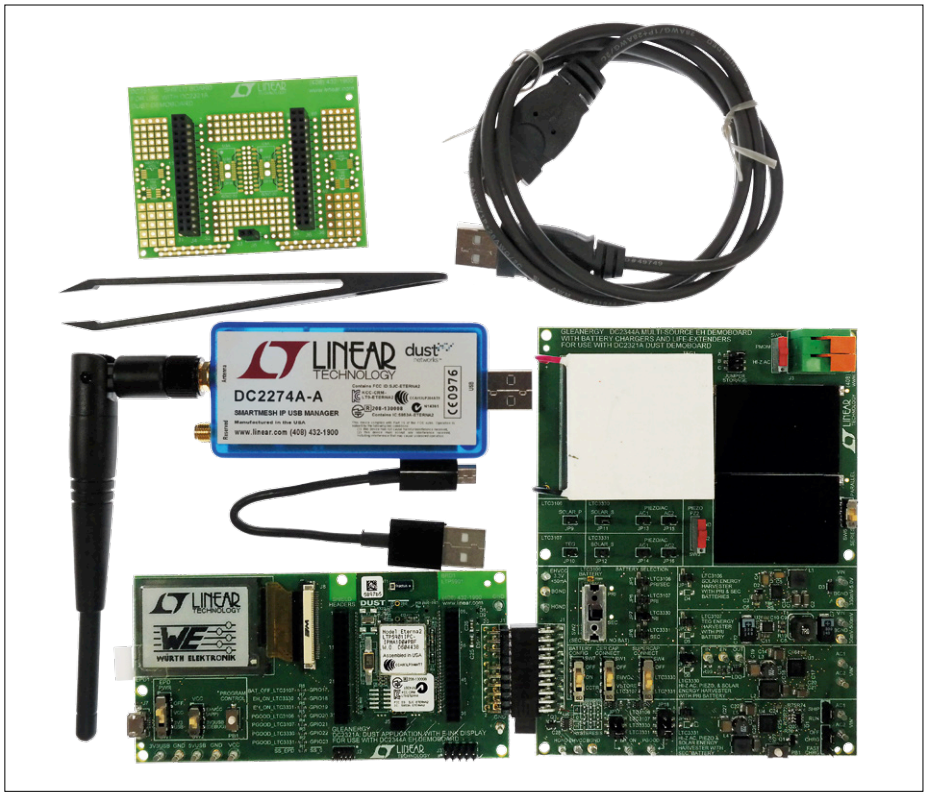


Bild 14. Man bekommt einiges für sein Geld: die zwei Platinen, hochwertige Kabel, den USB-Empfänger mit Antenne, einen Erweiterungs-Aufsatz und sogar eine Pinzette.

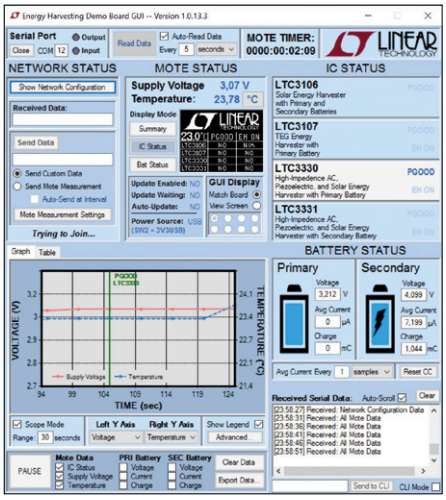


Bild 15. Das Tool QuikEval ist ganz auf die Entwicklung von Energy-Harvesting-Applikationen ausgerichtet.

(„secondary battery“), ein Supercap-Kondensator, ein Peltier-Element, zwei Solarzellen und die entsprechenden Treiberschaltungen mit vielen (Jumper-)Einstellungen, Mess-Abzweigungen und Ausgängen zu finden. Im Mittelpunkt stehen vier Regel-ICs von Linear Technology. Zwei davon sind universell für den Anschluss einer Solarzelle, eines Piezoelements oder eines Generators gedacht. Der größte Unterschied liegt in der zu ladenden Zelle: Der LTC3330 unterstützt die Batterie, der LTC3331 funktioniert mit dem Akku. Dann gibt es noch den LTC3107 für das Peltier-Element und Batterie sowie den LTC3106 nur für die Solarzelle, allerdings auch für beide Energiespeicher (Batterie/Akku). Durch die Jumper und Schalter auf der Vorderseite der Platine und die Brückenwiderstände auf der Rückseite sind unzählige Beschaltungs-Einstellungen möglich. Auch können einzelne Regler ein-, aus- oder parallelgeschaltet werden. Auf all dies kann hier leider nicht eingegangen werden. Die zugehörige Dokumentation erläutert aber das Board und dessen Möglichkeiten sehr penibel.

Das zweite Board kann direkt seitlich an das DC2344A gesteckt werden und heißt DC2321A [19]. Dieses Board besteht vor allem aus dem Dust-Mote-Modul LTP5901 sowie einem E-Ink-Display. Das LTP5901 ist ein Vertreter der Dust-Smart-Mesh-IP-Wireless-Familie von LT, arbeitet ähnlich dem bekannten Zigbee auf dem 2,4-GHz-Band nach IEEE 802.15.4e und besitzt einen ARM-Cortex-M3-Kern. Mit der Dust-Technologie können verschiedenartig strukturierte Funk-Netzwerke aufgebaut werden (Stichwort IoT). Im Gleanergy hat das LTP5901-Modul die Aufgabe, das Energy-Harvesting zu überwachen (wobei es natürlich auch selbst nach dem Low-power-Konzept versorgt wird) und die Daten seriell an der Micro-USB-Schnittstelle beziehungsweise auf dem E-Ink-Display auszugeben. Dazu gibt es ein Demo-Tool, mit dem man alle wichtigen Einstellungen der Regler überwachen kann, um letztendlich von der Prototyp-Phase zum eigenen Energy-Harvesting-Design zu kommen (**Bild 15**). Mit dem mitgelieferten leeren Shield DC2510A kann man die Schaltung erweitern. Die gesamte Lösung lässt sich natürlich auch kabellos betreiben, indem man die SmartMesh-Technologie mit dem DC2274A-A-USB-Stick nutzt. Auch dazu gibt es ein entsprechendes SDK (**S**oftware **D**evelopment **K**it). Insgesamt lohnt sich also die Anschaffung, wenn man nach Wireless- beziehungsweise E-Harvesting-Lösungen für eigene Anwendungen sucht und die Prototyp-Phase verkürzen will. ⏮

(160287)

Weblinks

[1] www.elektormagazine.de/150687

[2] www.infineon.com/cms/en/product/promopages/sensors-2go/

[3] goo.gl/S17DDt

[4] www.st.com/en/evaluation-tools/32f412gdiscovery.html

[5] www.atmel.com/tools/attiny817-xpro.aspx

[6] start.atmel.com

[7] www.cypress.com/products/psoc-analog-coprocessor

[8] goo.gl/E0Q92g

[9] goo.gl/7jsViq

[10] www.renesas.com/en-us/products/synergy/book.html

[11] synergygallery.renesas.com/

[12] goo.gl/d9OSpq

[13] goo.gl/RhxvQH

[14] goo.gl/RhZ0Dd

[15] goo.gl/au8na5

[16] goo.gl/lw4kaA

[17] www.linear.com/solutions/5744

[18] www.linear.com/solutions/7717

[19] www.linear.com/solutions/7662

IoT-Gateway und drahtlose Endknoten

Teil 1: Die Hardware

Weil Domotik von der Stange nicht immer die Funktionalität bieten kann, die ein maßgeschneidertes System verspricht, wurde der Autor zum Entwickler. In seinem System kommunizieren die Netzknoten (Nodes) drahtlos mit einem zentralen Gateway. Das Gateway nutzt das Protokoll MQTT, um Messdaten an einen OpenHAB-Server zu übergeben, der diese verarbeitet und visualisiert. Das Projekt wurde inzwischen auch von anderen Anwendern realisiert und erweitert. Für Elektor beschreibt der Autor in einem zweiteiligen Beitrag die wichtigsten Aspekte. Detaillierte Informationen sowie die zugehörige Software sind im Netz frei verfügbar.

Von **Hennie Spaninks** (NL)

Systeme zur Hausautomatisierung beflügeln die Erfindungsgabe vieler Konstrukteure und Entwickler. Faszinierend ist immer wieder, wenn ein Besucher vor der heimischen Haustür an einen fernen Standort gemeldet wird oder die Heizung im Winter die Gewohnheiten der Hausbewohner kennt. Domotik-Systeme gibt es inzwischen viele, doch sie haben meistens den einen oder anderen Schwachpunkt:

- Die Informationen bewegen sich auf Einbahnstraßen. Der Anwender kann zwar Kommandos senden, erfährt aber nicht, ob und wie sie ausgeführt wurden. Im Urlaub fern der häuslichen Umgebung, um nur ein Beispiel zu nennen, kann von verlässlichen Rückmeldungen vieles abhängen.
- Die Systemkomponenten lassen sich oft nur mit spezifischen Apps der Hersteller bedienen, die nicht miteinander kompatibel sind. Die Vielzahl der Apps sorgt auf dem Smartphone für Verwirrung.
- Ein Highlight der Domotik ist die Automatisierung: Aktionen können als Reaktionen auf Sensorwerte ausgeführt werden. Da die meisten marktgängigen Systeme nicht miteinander kompatibel sind, muss eine externe Instanz (beispielsweise *IfThisThenThat*) installiert werden, damit so ein System funktionsfähig ist.
- Viele drahtlose Systeme basieren auf WLAN. Für die zahllosen Wohngebäude aus Stahlbeton bedeutet dies, dass die Reichweite eingeschränkt ist.
- Datensicherheit wird nicht immer großgeschrieben. Die Gefahr ist real, dass Unbefugte auf sensible Daten zugreifen und die Anlage vom Nachbarn steuerbar ist.

Das alles war Grund genug, über ein eigenes, selbst entworfenes Domotik-System nachzudenken!

Erste Übersicht

Die Sensoren und Aktoren, hier *End Nodes* oder Endknoten genannt, sollen möglichst bezahlbar sein und sparsam mit der Energie umgehen. Die zentrale Steuerung muss natürlich die verbreiteten Netzwerkprotokolle wie TCP/IP beherrschen. Deshalb haben wir uns für ein zweistufiges System entschieden, das aus den Endknoten, dem Gateway

und einer zentralen Steuerung besteht. In **Bild 1** ist das Gesamtsystem schematisch dargestellt.

- Die Kommunikation mit den Endknoten findet duplex über drahtlose Übertragungswege auf vergleichsweise niedrigen Frequenzen statt. In Europa kommen das 433-MHz-Band und das 868-MHz-Band in Betracht. Da das 433-MHz-Band stark belegt ist, fiel unsere Entscheidung für das 868-MHz-Band. Der Hersteller *HopeRF* produziert für dieses Band Transceiver, die über einen SPI-Bus gesteuert werden können. Wir haben die Version RFM69HW gewählt, sie hat die höchste Sendeleistung. Diese Version ist mit einer Hardware-Verschlüsselung ausgerüstet, so dass die Datensicherheit gewährleistet ist.
- Einen TCP/IP-Stack besitzt das Modul RFM69HW nicht, es tauscht über die Funkverbindung feste Datenblöcke mit 66 Bytes Länge aus. Deshalb wird ein Gateway benötigt, das die Daten vom Drahtlosnetzwerk nach TCP/IP umsetzt, und umgekehrt.
- Um die Daten vom „Produzenten" (Sensorknoten) zum „Abnehmer" (beispielsweise ein Smartphone) zu übertragen, wird ein anderes verbreitetes Protokoll genutzt, das auf TCP/IP aufsetzt: MQTT. Dies ist ein Messaging-Protokoll, das entwickelt wurde, um Nachrichten auch an mehrere Abnehmer verbreiten zu können. Da MQTT ein etablierter Standard ist, können Datenströme des Domotik-Systems auch für andere, auf MQTT basierende Systeme genutzt werden.
- MQTT erfordert einen Server, der als *Broker* (Nachrichtenzentrale) fungiert. Abnehmer, beispielsweise Smartphones, können dort Nachrichten abonnieren. Realisiert wird der MQTT-Broker hier mit Mosquitto [1], eine Open-source-Implementierung, die für diverse Plattformen verfügbar ist, unter anderem auch für den Raspberry Pi.
- Unsere Domotik-Zentrale muss möglichst auch mit Systemkomponenten anderer Hersteller zusammenarbeiten und deshalb auch andere Protokolle unterstützen. Hier kommt OpenHAB [2] ins Spiel. Diese Open-source Software kann Nachrichten unterschiedlicher Protokolle in einen universellen Nachrichtenstrom umsetzen.
- Zu OpenHAB gehört eine Standard-App (Android und iOS), die über das Netz in einem sicheren Modus mit dem installierten OpenHAB-System kommuniziert. Überlegungen zu Firewalls oder Verschlüsselung der Verbindung zwischen Smartphone und Domotik-System entfallen. Über ein Web-Interface ist OpenHAB im lokalen Netzwerk zu erreichen.
- OpenHAB wird im Hintergrund von einer Datenbank gestützt, so dass unkompliziert Grafiken aus Messdaten erstellt werden können. Auch das Erstellen von Regeln und Skripts für eine Automatisierung ist nicht schwierig. OpenHAB besitzt auch externe Schnittstellen für Alarmmeldungen über E-Mail oder die OpenHAB-App.
- Zum Einsatz kommen preisgünstige und überall erhältliche Komponenten. Als Gehirn für die drahtlosen Endknoten wird ein Arduino verwendet, und ein Raspberry Pi dient als Plattform für Mosquitto und OpenHAB.

Protokoll für die Endknoten

Es stehen Endknoten für unterschiedliche Funktionen zur Verfügung. Vorhanden sind Sensoren für Temperatur und Feuchte, Schalter, PIR-Sensoren (die ausgelesen werden) sowie Relais und LC-Displays (die zu steuern sind). End Nodes können selbstständig in regelmäßigen Intervallen Messdaten absetzen, sie können die Messdaten aber auch auf Anforderung (nach einem Lesekommando) ausgeben.

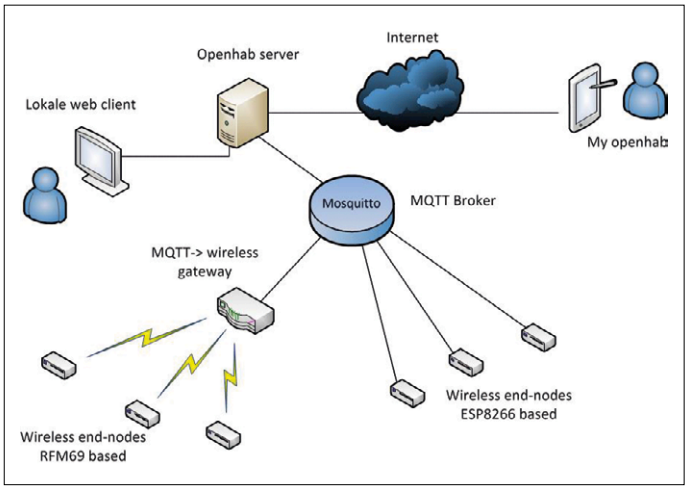


Bild 1. Struktur des Domotik-Systems.

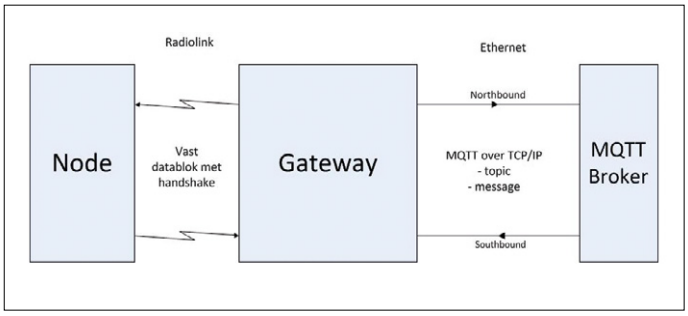


Bild 2. Hier sind die Datenströme skizziert, die durch das Gateway fließen.

Tabelle 1. Device IDs.

DevID-Bereich	Funktion
0…15	System Devices
16…31	Binärer Ausgang (Relais, LED)
32…39	Integer Output Devices (Dimmer, PWM)
40…47	Binärer Eingang (Schalter, PIR-Modul)
48…63	Dezimaler Eingang (Temperatur, Feuchte)
64…71	Integer Eingang (Tastatur, Schalter)
72	String Transfer (LC-Display, RFID-Leser)
73…90	Reserve für zukünftige Entwicklungen
90…99	Fehlermeldungen

Der Datenstrom, der durch das Gateway fließt, ist schematisch in **Bild 2** skizziert. Die Controller der Endknoten sind für unterschiedliche Funktionen programmierbar. An der *Device ID* (*DevID*), einem Schlüssel für die unterschiedlichen Knotentypen, erkennt das Gateway, wie die angelieferten Daten verarbeitet werden müssen. In **Tabelle 1** steht eine Übersicht über die DevIDs, die vom Gateway erkannt werden. Device 0 bis 15 sind Systemfunktionen, sie sind in **Tabelle 2** beschrieben. Die

Tabelle 2. Systemfunktionen.			
DevID	**Name**	**RW**	**Funktion**
00	Uptime	R	Minuten seit Start des Endknotens
01	TxInterval	RW	Sendeintervall in Sekunden (0 = keine periodische Übertragung)
02	RSSI	R	Feldstärke des Funksignals
03	Version	R	Software-Version des Endknotens
04	Voltage	R	Batteriespannung
05	ACK	RW	Flag für Bestätigung gesendeter Kommandos
06	Toggle	RW	Flag für Toggle-Funktion des Tasters auf dem Endknoten
07	Timer	RW	Flag für Timer-Funktion des Tasters auf dem Endknoten
08	Btnpress	RW	Flag für Senden der Nachricht „Taster gedrückt“
09	TXreply	R	Anzahl nötiger Wiederholungen während der Funkverbindung

Tabelle 3. Fehlermeldungen.		
Error ID	**Name**	**Beschreibung**
90	Link error	Die Funkverbindung ist unterbrochen.
91	Syntax error	Die MQTT-Nachricht enthält Syntaxfehler.
92	Invalid device	Die adressierte DevID existiert nicht im Endknoten.
99	Wakeup	Nachricht, die beim Starten des Endknotens gesendet wird.

Tabelle 4. MQTT-Beispiele.		
Topic	**Message**	**Beschreibung**
home/rfm_gw/sb/node02/dev16	ON	Schaltet die LED auf Node 2 ein.
home/rfm_gw/sb/node02/dev16	READ	Fragt den Status der LED auf Node 2 ab.
home/rfm_gw/sb/node03/dev01	300	Setzt das Tx-Intervall von Node 3 auf 5 min.
home/rfm_gw/sb/node03/dev01	0	Schaltet periodisches Senden auf Node 3 aus.
home/rfm_gw/sb/node18/dev48	READ	Liest die Temperatur von Node 18.
home/rfm_gw/sb/node05/dev02	READ	Liest die Feldstärke des Funksignals auf Node 5.
home/rfm_gw/sb/node05/dev03	READ	Liest die Software-Version von Node 5.

Fehlermeldungen, die das Gateway erzeugt, fasst **Tabelle 3** zusammen.

MQTT-Nachrichten

Das MQTT-Protokoll arbeitet mit „Abonnements“, sogenannten *Topics* (siehe auch [3]). Wenn ein Abnehmer auf ein Topic abonniert ist, sorgt der Broker dafür, dass er alle Nachrichten dieses Topics erhält.

Das verwendete Topic hat folgendes Format:

`home/rfm_gw/direction/nodeID/deviceID`

Darin gibt `direction` die Richtung des Datenstroms an, wobei `nb` (*Northbound*) für Sensordaten vom Node zum OpenHAB-Server und `sb` (*Southbound*) für Kommandos von OpenHAB zum Endknoten steht.
Die `nodeID` ist die ID des Knotens. Jeder Knoten besitzt eine

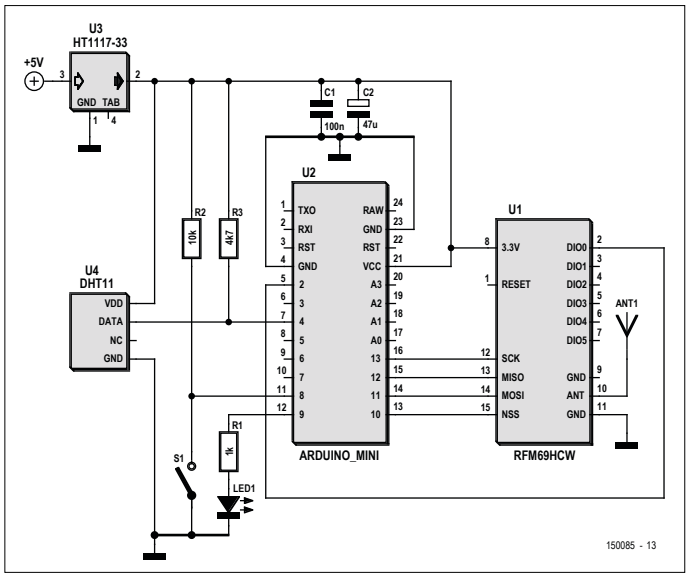


Bild 3. Die Hardware eines Endknotens.

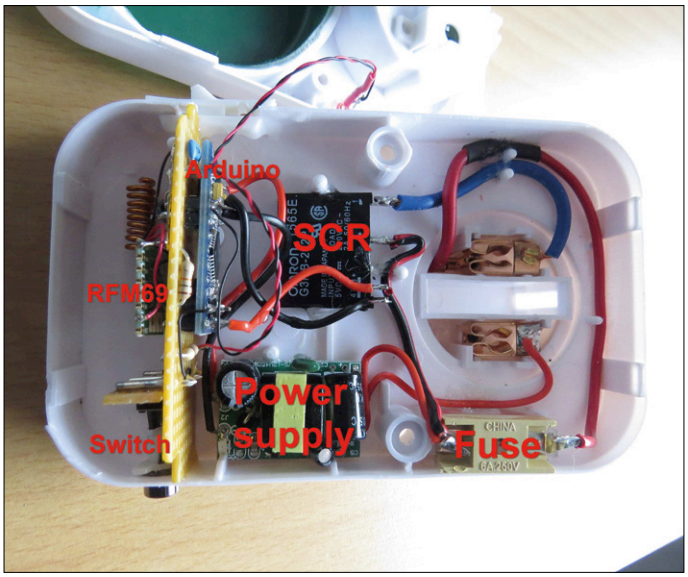


Bild 4. Ein DIG-Endknoten in einem Schaltuhrgehäuse.

einmalige ID (Adresse), sie wird während der Kompilierung des Codes zugewiesen. Das Gateway hat immer die `nodeID` 01. Die `deviceID` ist die ID des Knotentyps, wie zuvor beschrieben. Die MQTT-Nachricht (*payload*) hängt vom Knotentyp und der Richtung ab:

`Southbound:` Die Nachricht enthält Kommandos für den Knoten (ON, OFF, READ) oder Werte, die bei den Knoten gesetzt werden müssen (zum Beispiel die Anzahl der Sekunden für das Sende-Intervall).

`Northbound:` Die Nachricht enthält die Daten des betreffenden Parameters, der zum Knotentyp (DevID) gehört.
Das Gateway abonniert sämtliche Southbound-Nachrichten für alle Endknoten im Netzwerk mit einer Wildcard über das folgende Topic:
`home/rfm_gw/sb/#`
Der OpenHAB-Server soll sämtliche Nachrichten empfangen, die von den Endknoten kommen. Das Topic lautet deshalb:
`home/rfm_gw/nb/#`

Einige Beispiele für MQTT-Nachrichten enthält **Tabelle 4**.

Hardware der Endknoten

Die Endknoten basieren auf einem Arduino. Der Entwurf berücksichtigt folgende Vorgaben:

- Die Betriebsspannung des Drahtlosmoduls RFM69HW beträgt 3,3 V. Die Spannungen an den Eingängen dürfen 3,9 V nicht übersteigen. Deshalb kommt der Arduino Pro Mini zum Einsatz, der ebenfalls mit 3,3 V arbeitet. Eine Alternative ist der Arduino Buono R3, er ist auf 3,3 V umschaltbar.
- Das Drahtlosmodul hat beim Senden kurzzeitig einen Strombedarf von etwa 130 mA, die Stromversorgung muss entsprechend dimensioniert sein.
- Die Kommunikation zwischen dem Arduino und dem Drahtlosmodul verläuft über den SPI-Bus. Hierzu werden die Standard-Anschlüsse des Arduino benutzt.
- Das Drahtlosmodul ist ohne Antenne nicht betriebsfähig. In den meisten Fällen genügt ein Stück Draht, Länge 8,6 cm.

Die Schaltung eines Endknotens ist aus **Bild 3** ersichtlich. Dieser Node ist mit einem kombinierten Temperatur- und Feuchte-Sensor DHT11 bestückt, er ist mit Pin 4 des Arduino verbunden. Ein Taster liegt an Pin 8 und eine LED an Pin 9.

Über den SPI-Bus (SCK, MOSI, MISO und NSS) ist das Funkmodul an den Arduino gekoppelt. Die Betriebsspannung 3,3 V stellt ein Spannungsregler AMS1117 bereit, die nachgeschalteten Kondensatoren fangen Stromspitzen auf.
Die Software, die in den Arduino geladen wird, bestimmt die Funktion des Endknotens. Zurzeit sind die folgenden End Nodes verfügbar:

- DHT ist ein Endknoten mit Sensoren für Temperatur und Feuchte. Der Endknoten hat einen Taster und einen digitalen Ausgang.
- DIG ist die einfachste Version eines Endknotens. Er besitzt lediglich einen Taster und einen digitalen Ausgang.

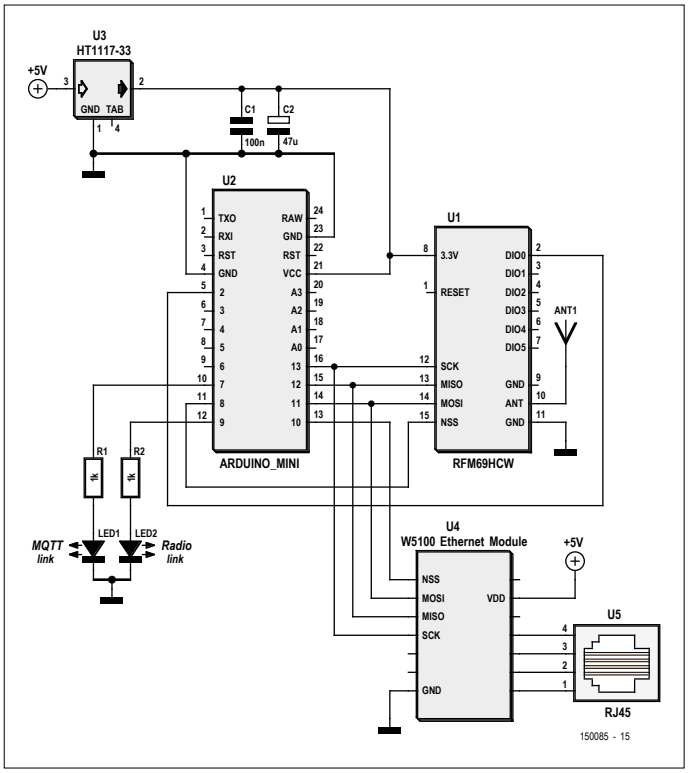


Bild 5. Die Hardware des Gateways.

- RFID ist ein Endknoten mit einem RFID-Leser. Wenn ein RFID-Tag erkannt wird, überträgt der Knoten die RFID-ID zum OpenHAB-Server.
- LCD ist ein Endknoten mit einem LC-Display. Von OpenHAB können Textzeilen zum Display übertragen werden.
- RC ist ein Endknoten, der mit einem Drahtlosmodul auf 433 MHz arbeitet. Über diesen Endknoten können Funkschalter des Home-Automation-Systems *KlikAanKlikUit* [4] betätigt werden.

Die Software ist bequem anpassbar, wenn Funktionen modifiziert oder hinzugefügt werden sollen. Der Aufbau eines Endknotens hängt in hohem Maß von der Funktion ab. So lässt sich der Node DIG im Gehäuse einer preisgünstigen mechanischen Schaltuhr unterbringen. Das Foto in **Bild 4** zeigt das Ergebnis.

Das Gateway

Die Komponenten des Gateways sind der Arduino, das Drahtlosmodul RFM69HW und ein Ethernetmodul W5100. Die Schaltung ist in **Bild 5** dargestellt. Auch im Gateway arbeitet ein Arduino, dessen Betriebsspannung 3,3 V beträgt.
Aus der Schaltung geht hervor, dass sich das Drahtlosmodul und das Ethernetmodul den SPI-Bus teilen. Die Leitungen SCK, MOSI und MISO sind parallel angeschlossen. Um das jeweilige Modul auszuwählen, werden unterschiedliche Slave-select-Signale (SS) benötigt. Das Drahtlosmodul ist mit Pin 8 des Arduino verbunden, das Ethernetmodul liegt an Pin 10.
Auch im Gateway sorgt ein Spannungsregler AMS1117 für die Betriebsspannung 3,3 V, zwei LEDs signalisieren den Status des Gateways.

Für den Ethernet-Anschluss kann ein eigenständiges Modul eingesetzt werden. Wenn der Arduino ein „Buono“ ist, ist ein

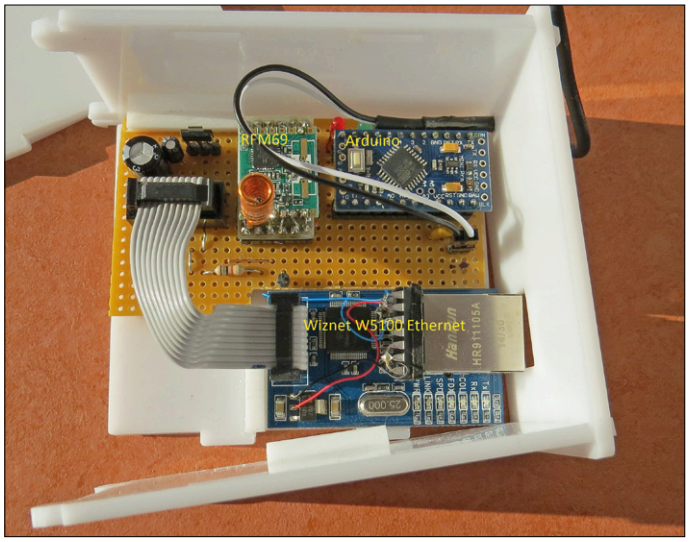


Bild 6. Das vom Autor aufgebaute Gateway.

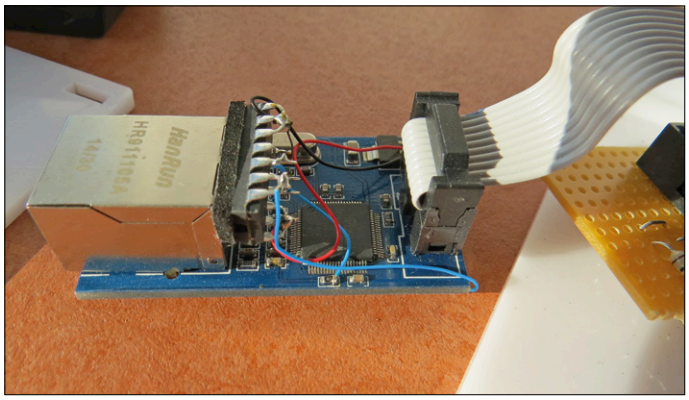


Bild 7. Der 4011 ist am Gehäuse des Netzwerkanschlusses befestigt.

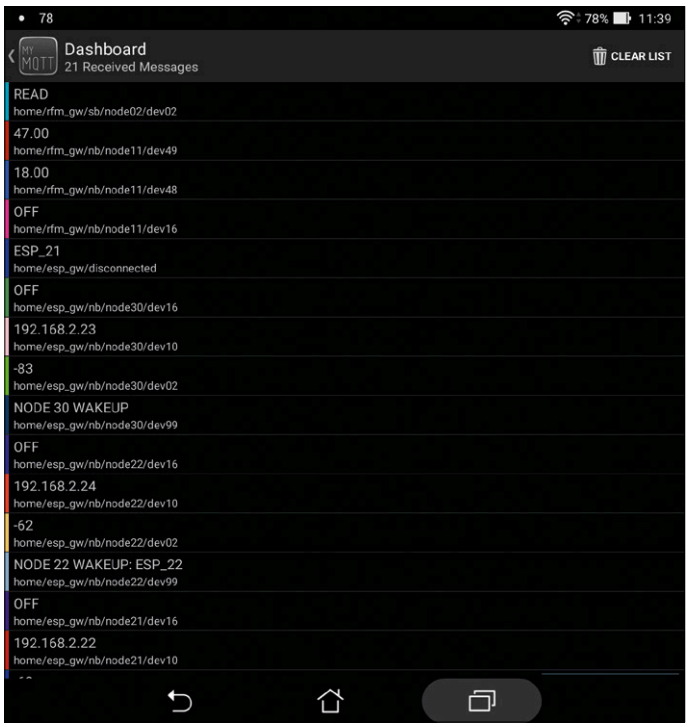


Bild 8. Beispiel für den MQTT-Nachrichtenaustausch.

Ethernet-Shield des Herstellers WIZnet eine mögliche Wahl. Für jedes Domotik-System wird nur ein Gateway benötigt. Der Autor hat deshalb das Gateway, wie **Bild 6** zeigt, auf einem Prototyp-Board aufgebaut.

Beim Aufbau des Gateways sind zwei Punkte zu beachten:

W5100-Bug

Das Ethernetmodul W5100 von WIZnet eignet sich nicht unmittelbar für den Einsatz mehrerer Module am SPI-Bus. Wenn das W5100 den SPI-Bus nicht belegt, gibt es ihn nicht selbstständig für andere Busteilnehmer frei. Das Problem wird durch einen Inverter gelöst, der das SEN-Signal aus dem CS-Signal ableitet (siehe [5] für weitere Informationen).
Bei neueren Versionen des W5100-Shields ist dieser Inverter bereits vorhanden. Das separate Modul, das der Autor auf eBay erwarb, besaß diesen Inverter nicht. Aus **Bild 7** ist ersichtlich, dass als Inverter ein IC (ein 4011) dient. Das Bild lässt auch erkennen, dass das Signal SEN unkompliziert über einen Pullup-Widerstand an die Platine angeschlossen werden kann. Die freien Eingänge des 4011 müssen an Masse gelegt werden!

Kurzschluss über die ICSP-Kontaktleiste

Wenn ein Arduino im Standard-Format (wie der Buono) zusammen mit einem W5100-Shield zum Einsatz kommt, wird dem Arduino über die ICSP-Kontaktleiste durch das Ethernet-Shield die Spannung 5 V zugeführt. Die Betriebsspannung 3,3 V des Arduino wird auf 5 V hochgezogen, so dass das Drahtlosmodul RFM69HW beschädigt werden kann. Das lässt sich verhindern, indem der VCC-Pin der ICSP-Kontaktleiste abgekniffen oder nach außen gebogen wird.

Wird fortgesetzt...

Im nächsten Teil dieses Beitrags geht es um die Software für das Gateway und die Endknoten sowie um die Konfiguration von Mosquitto und OpenHAB. Wer nicht so lange warten möchte, kann die Software vorab herunterladen [6]. Um das Gateway zu testen, muss Mosquitto vorhanden sein. Ein MQTT-Client kann den Test unterstützen. Unter Android ist die App *MyMQTT* gut brauchbar, und unter Windows sind *Chrome Lens* oder *MQTT.FX* geeignet. Einen Eindruck von den zu erwartenden Nachrichten gibt **Bild 8**.

Dieses Projekt wurde inzwischen mehrfach realisiert. Erweiterungen, Probleme und Erfahrungen werden gegenwärtig in einem Forum [7] diskutiert. ⏮

(150085)gd

Weblinks

[1] https://mosquitto.org

[2] www.openhab.org

[3] www.elektormagazine.de/tags/mein-weg-ins-iot

[4] www.klikaanklikuit.nl

[5] http://john.crouchley.com/blog/archives/662

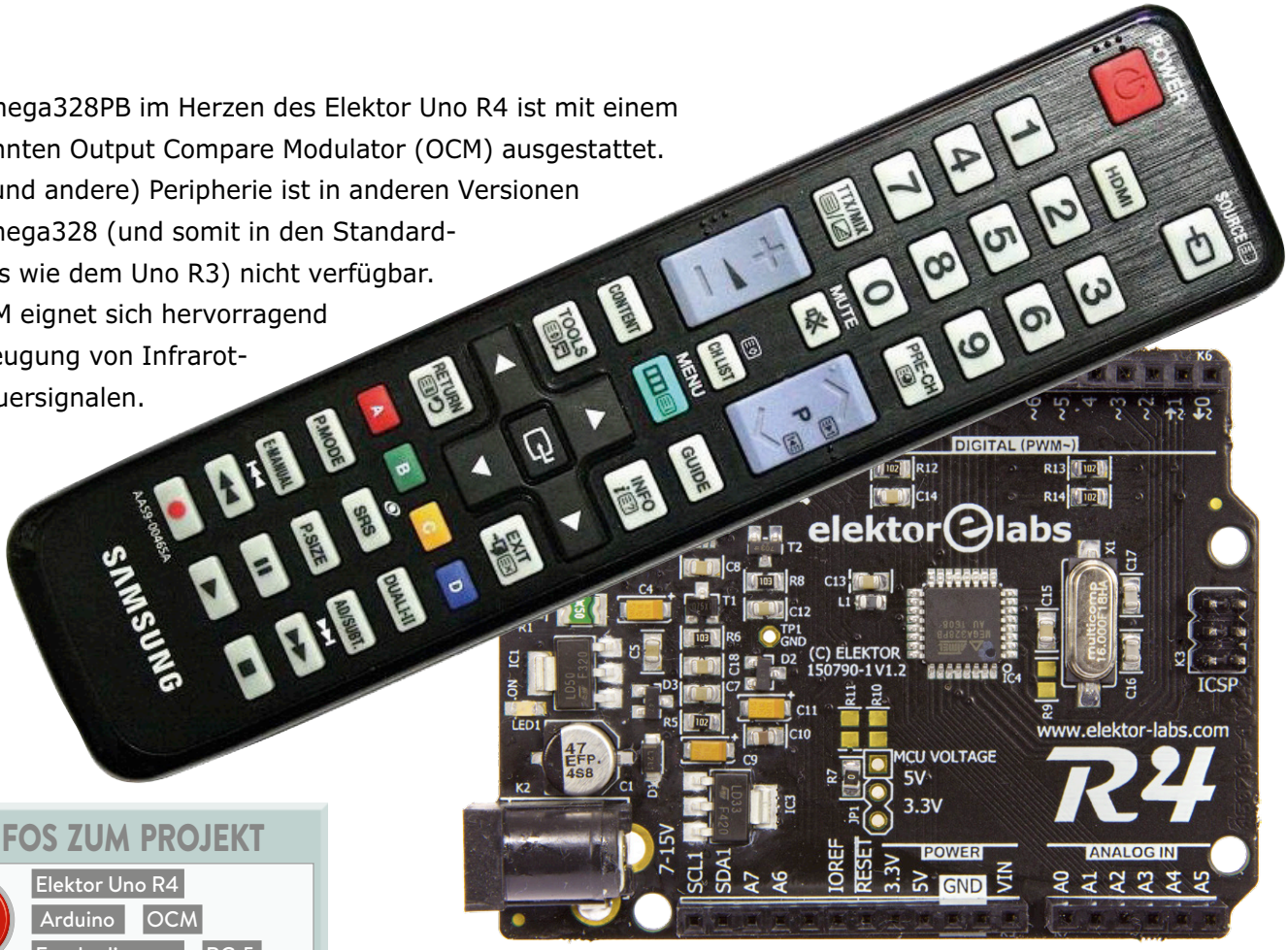
[6] https://github.com/computourist/RFM69-MQTT-client

[7] http://homeautomation.proboards.com/board/2/openhab-rfm69-based-arduino

NEC-Fernbedienungssignale

Exklusiv erzeugt mit dem Elektor Uno R4

Der ATmega328PB im Herzen des Elektor Uno R4 ist mit einem sogenannten Output Compare Modulator (OCM) ausgestattet. Diese (und andere) Peripherie ist in anderen Versionen des ATmega328 (und somit in den Standard-Arduinos wie dem Uno R3) nicht verfügbar. Der OCM eignet sich hervorragend zur Erzeugung von Infrarot-Fernsteuersignalen.



INFOS ZUM PROJEKT

- Elektor Uno R4
- Arduino
- OCM
- Fernbedienung
- RC-5

Einsteiger
➡ Fortgeschrittener
Experte

etwa 1 Stunde

Elektor Uno R4
Computer mit Arduino IDE

etwa 25 €

Von **Clemens Valens** (Elektor-Labor)

Der OCM kombiniert die Ausgänge von Timer3 und Timer4 - eine weitere exklusive Eigenschaft des Elektor Uno R4 - und stellt das Ergebnis an PD2 (Arduino-Pin 2 und MCU-Pin 32) bereit. Kombinieren bedeutet, wie **Bild 1** zeigt, entweder eine UND- oder ODER-Verknüpfung beider Signale. Der Wert des Registers PORTD2 bestimmt die Art der Verknüpfung: Bei einer „1“ wird der ODER-Modus gewählt, bei einer „0“ der UND-Modus. Signale wie in Bild 1 finden sich zum Beispiel bei Infrarotfernsteuerungen, wo sie den Strom durch eine IR-LED modulieren. Mein Zuhause ist (wie Ihres wahrscheinlich auch) mit Fernbedienungen (FBs) übersät. Vor einiger Zeit suchte ich nach einer RC-5-kompatiblen FB (altes Philips-Format), musste aber feststellen, dass bei aller Vielfalt kein Gerät das RC-5-Protokoll benutzte, nicht einmal mein Philips-DVD-Player (RC-6), sondern scheinbar meist eine Variante des japanischen NEC-Protokolls. Das ist verständlich, da die meisten ferngesteuerten Geräte in meinem Haus auch in Asien gefertigt wurden. Mit der schönen

OCM-Peripherie des ATmega328PB lassen sich NEC-kompatible FB-Signale spielend leicht selber produzieren.
Ein NEC-FB-Code beginnt mit einer Präambel, einem Burst von 9 ms, gefolgt von 4,5 ms Stille (eine Pause). Dann werden sukzessive die 32 Bits übertragen. Eine „0“ besteht dabei aus einem Burst und einer Pause von jeweils 562,5 µs (insgesamt 1125 µs) und eine „1“ aus einem Burst von 562,5 µs mit einer Pause von 1687,5 µs (macht 2250 µs). Die Sequenz muss durch die Postambel beendet werden, einen Burst, der das letzte Bit abschließt. Alle Zeitspannen in dem FB-Befehl betragen 562,5 µs oder ein Vielfaches davon. Im Folgenden soll „Impuls“ für die Dauer eines Bursts und „Periode“ für die Dauer eines Bursts plus Pause stehen.
Ein Burst ist ein hochfrequentes Trägersignal, das vom Modulator ein- und ausgeschaltet wird. Der Träger wird von Timer3 erzeugt und von Timer4 moduliert. Es würde auch anders herum funktionieren. Alle Timer im ATmega328 einschließlich Timer3 und Timer4 können Interrupts erzeugen, wenn sie überlaufen oder einen bestimmten Wert überschreiten. Durch diesen schönen Mechanismus lässt sich ein Träger mit den zu übertragenden Daten einfach modulieren: Bei jedem ausgelösten Interrupt wird das nächste zu sendende Bit aus dem TX-Puffer gelesen und bestimmt die Periode für eine „0“ oder eine „1“. Der Sketch zeigt, wie wir das erreichen können.

```
/*
 * IR Transmitter
 * Uses Output Compare Module (OCM), ATmega328PB
   only.
 */

#define CARRIER  38000
#define RC_CODE  0x6170807f

const int ir_led = 2;
const uint16_t pulse = 1125; // 2*562.5
uint32_t tx_buffer = 0;
uint32_t tx_index = 0;

#define TIMER3_ON  TCCR3B |= (1<<CS30) /* prescaler
   1 */
#define TIMER3_OFF  TCCR3B &= ~(1<<CS30)
#define TIMER4_ON  TCCR4B |= (1<<CS41) /* prescaler
   8 */
#define TIMER4_OFF  TCCR4B &= ~(1<<CS41)

ISR(TIMER4_COMPB_vect)
{
  if (tx_index!=0) // Any bits left?
  {
    OCR4B = pulse; // Set pulse length.
    if ((tx_buffer&tx_index)!=0)
    {
      OCR4A = OCR4B<<2; // Set '1' period.
    }
    else
    {
      OCR4A = OCR4B<<1; // Set '0' period.
    }
    tx_index >>= 1; // Next bit.
  }
  else
  {
    if (OCR4A>=2*OCR4B)
    {
      OCR4A = OCR4B + 10; // Send closing burst.
    }
    else
    {
      // Done, clean up.
      TIMSK4 &= ~(1<<OCIE4B); // Disable interrupts.
      TIMER3_OFF;
      TIMER4_OFF;
    }
  }
}

void ir_send_code(uint32_t code)
{
  // Copy code to TX buffer.
  tx_buffer = code;
  tx_index = 0x80000000; // 32 bits.
  // Setup timer4 for preamble.
  OCR4A = 2*12*pulse;
  OCR4B = 2*8*pulse;
  TIFR4 = (1<<OCF4B); // Clear pending interrupts.
  TIMSK4 = (1<<OCIE4B); // Enable compare interrupt.
  // Go!
  TIMER3_ON;
  TIMER4_ON;
}

void setup(void)
{
  pinMode(ir_led,OUTPUT); // Enable output.
  digitalWrite(ir_led,LOW); // OCM in AND mode.
  // Timer3 mode 4 (CTC), toggle output on match.
  TCCR3A = (1<<COM3B0);
  TCCR3B = (1<<WGM32);
  OCR3A = 16000000/CARRIER/2; // Toggle at 76 kHz ->
   38 kHz.
  OCR3B = OCR3A>>1; // 50% duty-cycle.
  // Timer4 mode 15 (Fast PWM), clear output on
   match.
  TCCR4A = (1<<WGM41) | (1<<WGM40) | (1<<COM4B1);
  TCCR4B = (1<<WGM43) | (1<<WGM42);
  ir_send_code(RC_CODE);
}

void loop(void)
{
}
```

Alles beginnt in der Funktion `setup` mit der Deklaration von Pin 2 (PD2, an dem die IR-LED angeschlossen wird) als Ausgang.

Wenn der Ausgang Low wird, ist nicht nur die LED ausgeschaltet, auch der OCM arbeitet dann im AND-Modus. Als nächstes richten wir Timer3 im CTC-Modus 4 (Compare-Modus) für eine Frequenz von 38 kHz ein, einer beliebten FB-Trägerfrequenz. Der Timer muss auf das Doppelte dieser Frequenz eingestellt werden, um das Toggeln des Ausgangspins bei jedem Überlauf zu kompensieren, wobei die Timerfrequenz effektiv durch zwei dividiert wird. Pin 2 ist der OC3B-Ausgang, der von OCR3B gesteuert wird. Wir setzen dieses Register auf den halben Wert von OCR3A, um ein Tastverhältnis von 50 % zu erreichen. Timer4 arbeitet im Fast-PWM-Modus 14, wobei OCR4A die Frequenz und OCR4B das Tastverhältnis bestimmen. Der Rest seiner Konfiguration wird bis zur Funktion `send_code` verzögert. Zunächst kopiert diese Funktion den zu übertragenden Code in den 32-Bit-TX-Puffer. Da ein Code mit einer Präambel beginnt, werden die entsprechenden Impuls- (9 ms) und Periodenwerte (13,5 ms) in die Impuls- (OCR4B) und Perioden-Register (OCR4A) geladen. Die Multiplikation mit zwei ist erforderlich, da der MCU-Takt 16 MHz beträgt und der Prescaler nur durch 1, 8, 64, 256 oder 1024, nicht aber durch 16 teilen kann. Da der Prescaler auf den Divisor 8 statt 16 eingestellt ist, müssen die Impuls- und Periodenwerte verdoppelt werden. Der nächste Schritt besteht darin, die Interrupts vorzubereiten und die Timer zu starten.

Die COMPB-ISR wird am Ende eines Impulses aufgerufen. Wenn dies geschieht und noch nicht alle Bits versandt worden sind, muss der Timer für eine Periode von entweder 1125 µs (wenn das zu sendende Bit eine „0“ ist) oder 2250 µs (bei einer „1“) neu konfiguriert und die Impulslänge auf 562,5 µs (identisch für „0“ und „1“) eingestellt werden. Wie zuvor werden alle Werte verdoppelt, um den „falschen“ Prescaler-Wert zu kompensieren. Der 32-Bit-Code wird durch einen abschließenden Burst beendet, ansonsten kann der Empfänger nicht erkennen, ob das letzte Bit eine „0“ oder eine „1“ war.

Wenn `tx_index` gleich null ist, sind alle Bits versendet. Wir können nun das OCR4A-Register verwenden, um zum Beispiel mit einem speziellen Wert einen endgültigen Burst zu senden, was auch bedeutet, dass die ISR ein weiteres Mal gestartet wird. Mit einem Wert kleiner als 1125 µs und größer als OCR4B wird der entsprechende Zustand beim nächsten Aufruf der ISR einfach detektiert. Natürlich ist es auch möglich, hierfür einen anderen Mechanismus (zum Beispiel mit einer statischen Variable) einzusetzen, aber warum nicht das ohnehin verfügbare Register verwenden? Als letztes nach der Präambel, den 32 Codebits und dem Stopburst schaltet die ISR die Lichter aus, will meinen, die Timer ab, da sie nicht mehr benötigt werden. Die Funktion `loop` bleibt leer, da der FB-Code nur einmal nach Drücken der Reset-Taste gesendet wird.

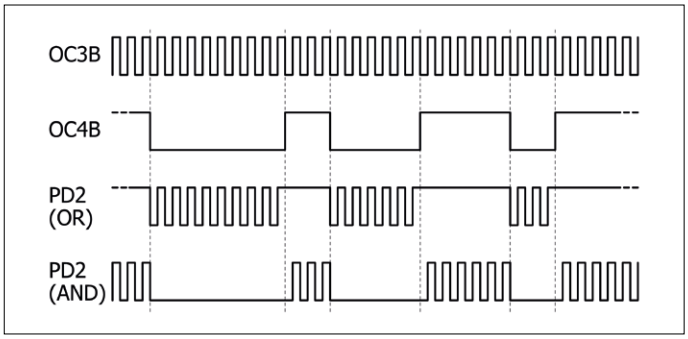


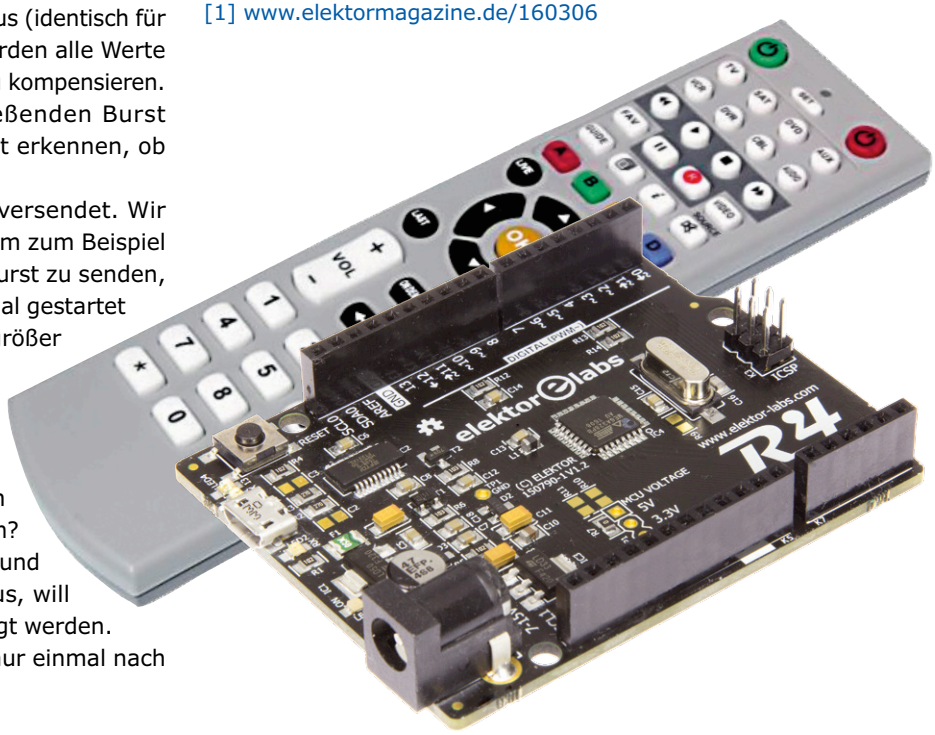
Bild 1. Im OCM-Modus erscheinen an PD2 entweder „OC3B OR OC4B“ (PORTD2 = 1) oder „OC3B AND OC4B“ (PORTD2 = 0).

Das Senden eines Codes ist vollständig ISR-gesteuert und läuft deshalb im Hintergrund, so dass der Rest des Programms sich mit anderen Dingen befassen kann. Das Programm kann das OCIE4B-Bit im TIMSK4-Register befragen, ob der Sender noch beschäftigt ist. Eine komplexe Programmierung ist nicht nötig, alles wird direkt von der Hardware-Peripherie abgehandelt. Der Nachteil dieser Einfachheit ist allerdings, dass das Ganze nur mit Pin 2 funktioniert. Aber das können wir locker in Kauf nehmen! ◀

(160306)

Weblink

[1] www.elektormagazine.de/160306



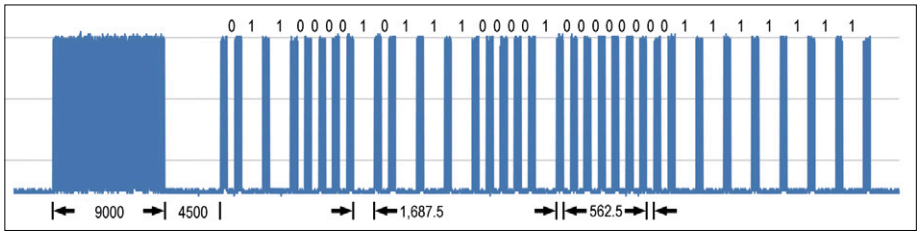


Bild 2. Ein Oszillogramm des vom Sketch erzeugten Signals an PD2.

EINKAUFSZETTEL
→150790-91:
Elektor Uno R4

OBD2-Handheld mit dem Raspberry Pi

Mit neuer Diagnosesoftware

Von **Thomas Beck**

Das Modul Pi-OBD-HAT ist ein Add-on-Board für den Raspberry Pi, es verwandelt den Kleinstcomputer in ein OBD2-Diagnoseinterface. Ein Nachteil beim Handling war bisher, dass man die Lösung per Terminalprogramm ansteuern musste. Wesentlich einfacher und komfortabler geht es mit der hier vorgestellten OBD2-Diagnosesoftware HHGui.

HHGui (PI-OBD Handheld GUI) ist eine für das Pi-OBD-HAT-Modul angepasste Variante meines OBD2-Handheld-Emulators HHEmu [1], einer Art Schweizer Taschenmesser für Entwicklungen rund um den „OBD2-Analyser NG/DIAMEX Handheld Open“ (HHOpen). Dies ist ein OBD2-Diagnosetester im Handheld-Format, der im September 2009 in Elektor veröffentlicht wurde [2].

Der OBD2-Analyser NG basiert auf dem DXM-OBD2 Modul [3], einem Vorgänger des AGV4000C-Interfaces im Pi-OBD-Modul.

Entwicklungsgeschichte

Der ungewöhnliche Weg der Softwareentwicklung zeigt, wie eine ursprünglich für einen Mikrocontroller geschriebene Firmware auf einen Raspberry Pi oder PC portiert werden kann - im Idealfall ohne den Sourcecode der Firmware zu verändern.

Ursprünglich war HHEmu ein reines Entwicklungswerkzeug, um Updates der AVR-Mikrocontroller-Firmware des OBD2-Analysers auf einem PC auszuführen und ohne Fahrzeug testen zu können. Deshalb enthält HHEmu neben der Firmware einen OBD2-Simulator, der simulierte OBD2-Steuergerätedaten über ein ebenso simuliertes DXM-OBD2-Modul an die Firmware liefert.
Im nächsten Entwicklungsschritt bekam HHEmu Zugriff auf serielle Schnittstellen eines PCs. Das eröffnet im Zusammenspiel mit der im April 2010 veröffentlichten Bluetooth-Erweiterung [4] für den OBD2-Analyser NG außergewöhnliche Möglichkeiten. Die auf einem PC innerhalb HHEmus ausgeführte Firmware konnte über die serielle Schnittstelle eines Bluetooth-Adapters mit der Bluetooth-Erweiterung eines echten NG-Analyzers verbunden werden und so das DXM-Modul steuern. Damit war der Test neuer Firmware an einem realen Fahrzeug möglich, ohne die Firmware zuvor in den OBD2-Analyser flashen zu müssen. Ein echter Meilenstein der Entwicklung! Ganz nebenbei ist HHEmu dadurch zur OBD2-Diagnosesoftware für DXM-basierte OBD2-Diagnoseinterfaces geworden. Zusätzlich konnte der OBD2-Simulator von HHEmu jetzt als Datenlieferant für OBD2-Diagnosesoftware anderer Hersteller oder für die Entwicklung neuer OBD2-Software fungieren. Neben der besseren Testmöglichkeit für die Firmware zeigte sich ein weiterer Vorteil der Portierung auf den PC. Bei jedem zukünftigen Firmware-Update erhält die Diagnosesoftware automatisch das funktionell identische Update.
Dann folgte die Portierung HHEmus auf den Raspberry Pi. Damit kann jetzt nach allen Anpassungen an das Pi-OBD-Modul und dem Entfernen OBD2-Analyser-NG spezifischer Menüs eine erste Version von HHGui veröffentlicht werden.

Betriebsarten

Aufgrund der Entwicklungsgeschichte ist HHGui in erster Linie eine OBD2-Diagnosesoftware, die sowohl mit ihrer Benutzeroberfläche das Aussehen des OBD2-Analysers NG imitiert als auch intern dessen an das Pi-OBD-Modul angepasste Firmware verwendet. Der aktuelle Funktionsumfang der Firmware ist dem **Kasten** zu entnehmen.

HHGui stellt seine Benutzermenüs und die OBD2-Daten in einem simulierten LCD-Display dar, das auf das größere Grafikdisplay des Raspberry Pi skaliert wird. Das ist ein Kompromiss, um neben dem offiziellen 7"-Touchscreen, der für den Raspberry Pi empfohlen wird, auch kleine Displays mit nur 320x240 Bildpunkten am Pi zu unterstützen. Allerdings ist dann ein Umbau des Pi-OBD-Moduls erforderlich, wie er im ersten Projektupdate des Labs-Projekts [5] beschrieben ist.
Als zweite Funktionalität ist im HHGui ein OBD2-Simulator implementiert, der das Pi-OBD-Modul und bis zu acht konfigurierbare OBD2-Steuergeräte simuliert. So ist es möglich, den vollen Funktionsumfang der Software auch ohne Pi-OBD-Modul und ohne Fahrzeug zu testen.
Da ein echtes Fahrzeug nur einen Teil aller möglichen OBD2-Dienste und Unterfunktionen unterstützt (ein großer Teil der Funktionen ist entweder nur für Benziner oder nur für Diesel-Fahrzeuge relevant), ist das eine einfache Möglichkeit, alle Funktionen auszuprobieren. Dazu ist diese Variante auch noch völlig gefahrlos, da nicht aus Versehen OBD2-Daten im Fahrzeug gelöscht werden können. Der OBD2-Funktionsumfang des Simulators ist identisch zum Funktionsumfang der Firmware. Die Tastenkommandos zur Steuerung des OBD2-Simulators wurden schon im Labs-Projekt zum HHEmu [1] beschrieben.

OBD2-Funktionsumfang der HHGui Diagnosesoftware

HHGui unterstützt die unten aufgeführten OBD2-Dienste nach ISO 15031-5. Eine genauere Beschreibung der Dienste findet sich auf der Projektseite [5], über die Unterfunktionen kann man sich bei [6], [7] oder [8] informieren. Die tatsächlich unterstützten Dienste und Unterfunktionen sind abhängig vom jeweiligen Fahrzeug beziehungsweise seinen OBD2-Steuergeräten. HHGui fragt die vom jeweiligen Steuergerät unterstützten Unterfunktionen ab und zeigt nur diese an.

- Unterstützt die laut ISO 15765-4 [9] maximal möglichen acht OBD2-Steuergeräte (Electronic Control Unit ECU)
- OBD2-Service 0x01: Momentanwerte lesen (Current Data); es werden folgende Unterfunktionen (Parameter Identifier PIDs) unterstützt: 0x00...0x60, 0x70, 0x80, 0x8D
- OBD2-Service 0x02: Fehlerumgebungsdaten lesen (Freeze Frame Data), unterstützte PIDs wie bei Service 0x01
- OBD2-Service 0x03: bestätigte Fehler lesen (Confirmed Diagnostic Trouble Codes DTC)
- OBD2-Service 0x04: Fehler, Fehlerumgebungsdaten, MIL-Status, OBD2-Monitor- und weitere Daten löschen
- OBD2-Service 0x07: schwebende Fehler lesen (Pending DTCs)
- OBD2-Service 0x09: Fahrzeuginformation lesen (Vehicle Information); es werden folgende Unterfunktionen (InfoTypes) unterstützt: 0x00, 0x02, 0x04, 0x06, 0x08, 0x0A, 0x0B
- OBD2-Service 0x0A: permanente Fehler lesen (Permanent DTCs)

Möglichkeiten der OBD2-Diagnose

Wenden wir uns nun wieder der Hauptfunktion OBD2-Diagnosesoftware zu. Was kann mit OBD2-Diensten im Fahrzeug erreicht werden und was nicht?
Wenn man die wichtigste Spezifikation ISO 15031-5 [6] (oder die inhaltlich identische SAE J1979 [7]) zu diesem Thema liest, wird klar, dass mit OBD2-Diensten nur emissionsrelevante Daten und Fehler gelesen und teilweise auch gelöscht werden können. Diese Daten werden hauptsächlich vom Motorsteuergerät geliefert. In Fahrzeugen mit Automatik-Getrieben liefert oft auch das Getriebesteuergerät OBD2-Daten. Seltener sind weitere Steuergeräte oder auch mehrere Motorsteuergeräte als Datenlieferanten anzutreffen, zum Beispiel in Hybrid-Fahrzeugen. Einstellungen wie etwa die zur automatischen Türverriegelung oder zur Abschaltung des Gongs der Gurtwarnung können mit OBD2-Diensten nicht vorgenommen werden. Bis auf die Motorkontrollleuchte können auch keine Fehlerlampen im Kombiinstrument gelöscht und weder Service-Anzeigen noch Service-Intervalle zurückgesetzt werden. Das alles ist

nur mit herstellerspezifischer Diagnose möglich, was deshalb nur für Fahrzeuge dieses einen Herstellers funktioniert.
Das ist zuerst einmal eine Enttäuschung. Der Vorteil der herstellerübergreifend standardisierten OBD2-Diagnose ist aber, dass sie heute, etwa 20 Jahre nach ihrer ersten Einführung in den USA (in Europa erst ab dem Jahr 2000 für Benziner), fast mit jedem Fahrzeug funktioniert. Seit 2008 hat sich auch die Vielzahl von Protokollen zur Übertragung von OBD2-Daten stark vereinheitlicht, da in Neufahrzeugen nur noch CAN-Protokolle für die OBD2-Datenübertragung zugelassen sind.
Wenn man sich dann die OBD2-Unterfunktionen genauer ansieht, wird man unglaublich viele Sensordaten, Zähler und andere Daten finden (insgesamt mehr als 100), die garantiert den eigenen Forscherdrang wecken. Außerdem sind wider Erwarten nicht nur Daten vom Motorsteuergerät dabei. Manche Daten werden zwar für die OBD2-Diagnose vom Motorsteuergerät geliefert, stammen aber ursprünglich von anderen Steuergeräten wie zum Beispiel dem Bremsensteuergerät.
Zu den einfach verständlichen OBD2-Daten zählen Daten, die bereits im Fahrzeug angezeigt werden, dort aber für die Anzeige aufbereitet, wie:

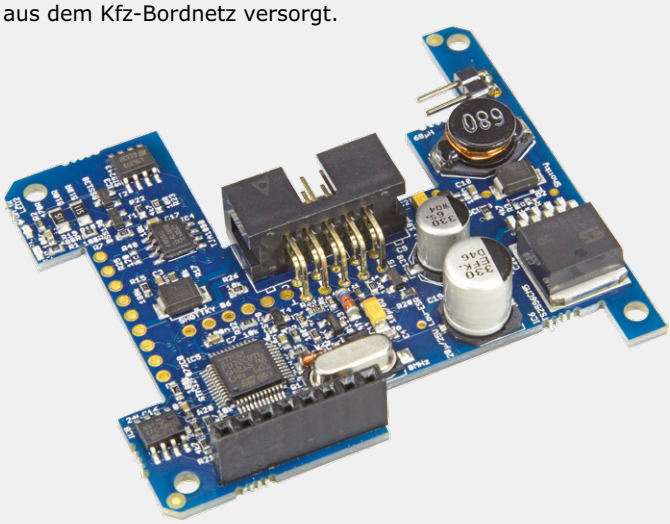
- mit einer Dämpfung (Außentemperatur, Tankfüllstand, generell alle Anzeigen mit Zeigerinstrumenten, die mit Schrittmotoren realisiert sind)
- mit einer Plateau-Funktion (Kühlmitteltemperatur)
- mit zusätzlich addierten Offsets, um gesetzliche Vorgaben der Mindestanzeige, Anzeigetoleranzen oder einfach Herstellerwünsche einzuhalten (Geschwindigkeit, Drehzahl)

Dafür liefert die OBD2-Diagnose noch nicht aufbereitete Sensordaten im Rohformat, also zum Beispiel die tatsächliche und nicht die angezeigte Geschwindigkeit.
Dann gibt es OBD2-Daten, die man gerne sehen würde, die aber der Hersteller nicht zur Anzeige vorgesehen hat, zum Beispiel die Öltemperatur oder den Turbolader-Ladedruck. Auch die Motorlast ist eine nützliche Information.
Ebenfalls für den Durchschnittsanwender relevant sind OBD2-Daten, die die Prüfbereitschaft für die nächste staatlich verordnete Fahrzeugprüfung (in Deutschland beim TÜV) anzeigen. Dazu müssen alle OBD2-Prüfprogramme (OBD2-Monitore) seit dem letztmaligen Löschen der Fehler mindestens einmal erfolgreich durchlaufen worden sein. Dieser Zustand ist erreicht, wenn HHGui im Inspection/Maintenance-Readiness-Menü für alle OBD2-Monitore den Status „… monitoring ready: YES“ anzeigt.

Neben diesen leicht verständlichen OBD2-Daten gibt es viele weitere aus dem Bereich der Lambdaregelung, die eher etwas für Spezialisten sind. Die Auflistung aller Daten ist hier zu lang. Da die oben angegebenen Standards in ihren neuesten Versionen nur für viel Geld zu erwerben sind, empfiehlt sich ein Blick in die Wikipedia [8].

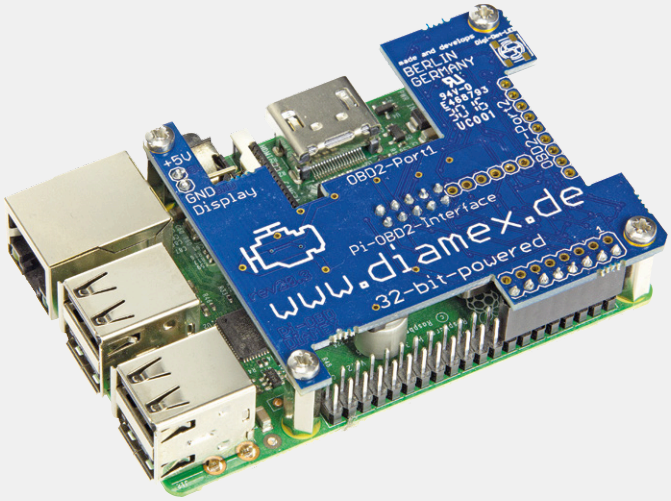
PI-OBD: Die Hardware

Das Modul PI-OBD-HAT [10] von Diamex verwandelt den Raspberry Pi in einen OBD2-Diagnoseadapter. Stattet man den Pi noch mit dem 7“-Touchdisplay aus, hält man ein leistungsfähiges Stand-Alone-Diagnosesystem für Kfz mit OBD2-Schnittstelle in den Händen. Dabei werden Raspberry Pi und Display aus dem Kfz-Bordnetz versorgt.



Für dieses OBD2-Diagnosegerät eignen sich im Prinzip alle Versionen des Raspberry Pi, allerdings sollte man wegen der höheren Prozessorgeschwindigkeit den Versionen Pi 2 Model B oder Pi 3 Model B den Vorzug geben. Das PI-OBD-Modul belegt nur acht Kontakte des Erweiterungssteckverbinders des Raspberry Pi, zur Stromversorgung, für die serielle Datenübertragung, für einen Reset und zur Aktivierung des PI-OBD-Bootloaders. Die Kommunikation erfolgt über die RxD- und TxD-Leitungen mit 115.200 Baud, Parameter 8N1.

Aufbau
Zunächst steckt man das PI-OBD-Modul auf den Raspberry Pi. Die 8-polige Buchsenleiste muss auf die äußeren Kontakte der



I/O-Stiftleiste des Pi gesteckt werden. Über die dem Modul beiliegenden Abstandsbolzen wird die Modulplatine fest mit dem Raspberry Pi verschraubt.

Der Raspberry Pi wird über die Buchsenleiste mit Strom versorgt, sobald der OBD2-Stecker am Fahrzeug angesteckt wird. Es ist außer beim Test und bei der Konfiguration keine eigene

Kommen wir nun zum interessanten Teil für Entwickler.

HHGui: Aufbau und Funktionsweise

Funktionsprinzip

Die Firmware wird in einem Thread ausgeführt, alle Eingaben und Ausgaben der Firmware werden durch weitere Threads verarbeitet und beantwortet. Dazu müssen Mikrocontroller-Register und Interrupts simuliert werden.

Tatsächliche Realisierung

HHGui besteht aus sechs Threads. **Bild 1** zeigt ein vereinfachtes Blockdiagramm mit den Interaktionen zwischen den Threads. Die tatsächlichen Namen der Threads in den Sourcedateien unterscheiden sich aufgrund der Entwicklungshistorie. Der Pi-OBD-Modul-Thread heißt dort DxmThread und der OBD2-Analyser NG-Firmware-Thread HhopenThread. Diese kürzeren Bezeichnungen werden ab jetzt auch im folgenden Text verwendet.

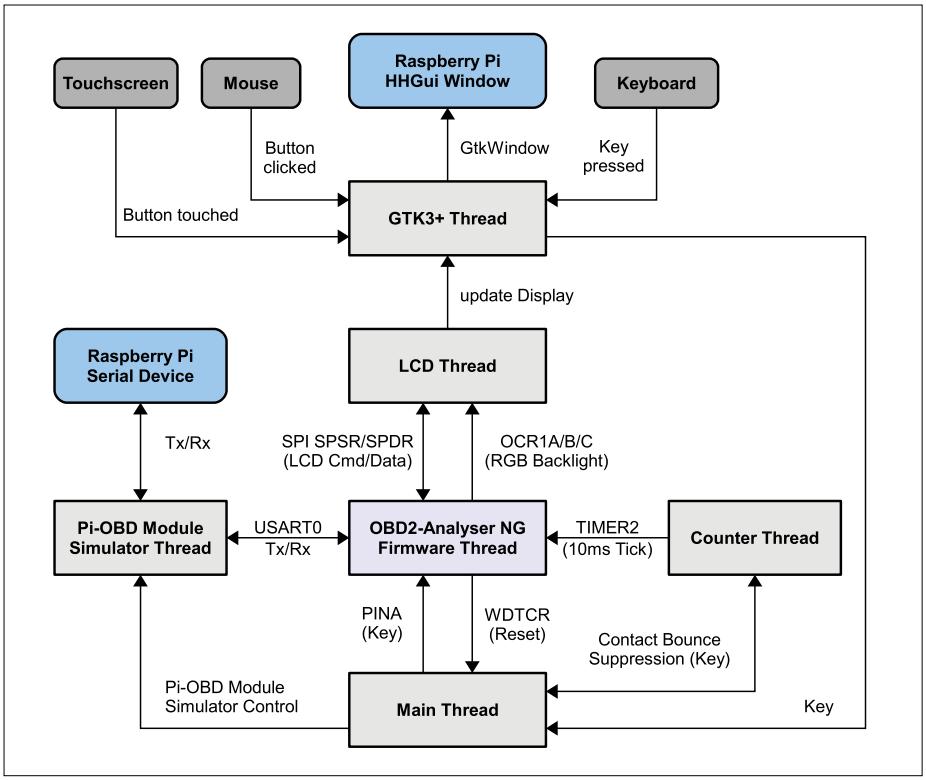
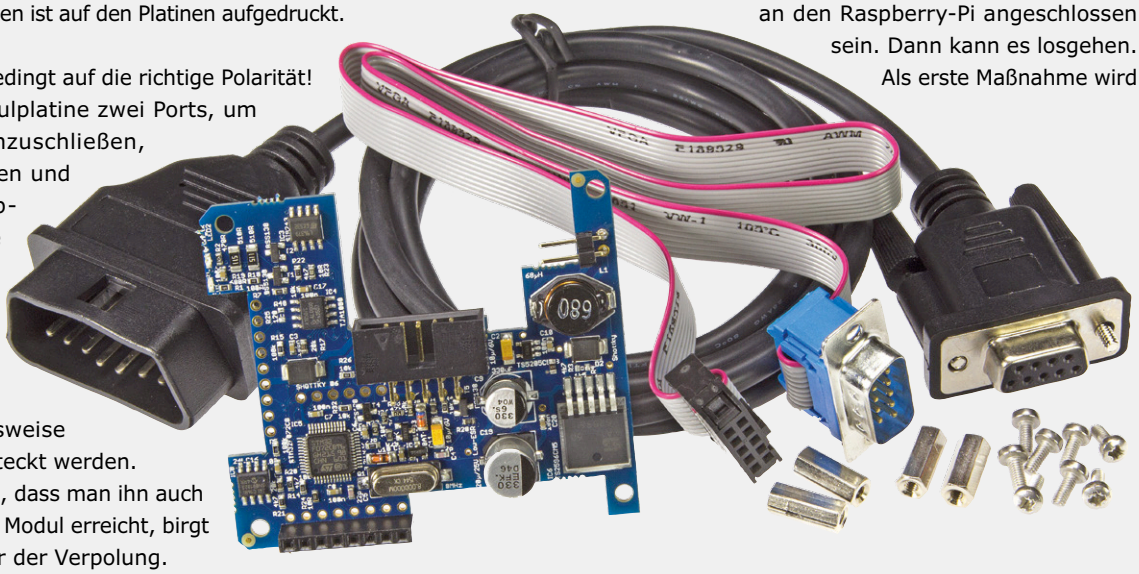


Bild 1. HHGui: Das Blockdiagramm.

Stromversorgung über die USB-Buchse erforderlich. Damit auch das 7"-Raspberry-Display mit Strom versorgt wird, muss dessen Verbindungskabel an die zweipolige Stiftleiste am Modul angeschlossen werden. Im Foto ist unten der Raspberry Pi mit aufgesteckten OBD-Modul und oben die Displayplatine zu sehen. Die Belegung der Stiftleisten ist auf den Platinen aufgedruckt.



Achten Sie bitte unbedingt auf die richtige Polarität! Es gibt auf der Modulplatine zwei Ports, um das OBD2-Kabel anzuschließen, für „lose" Kabelenden und für Kabel mit Sub-D-Stecker. Auf die 9-polige Stiftreihe von Port 2, die sich außen an der Modulplatine befindet, kann ein Flachstecker (beziehungsweise eine Buchse) aufgesteckt werden. Port 2 hat den Vorteil, dass man ihn auch bei aufgeschraubtem Modul erreicht, birgt aber auch die Gefahr der Verpolung.

Achten Sie auf den Platinenaufdruck! Beim Port 1, einer 9-poligen Buchse mit Orientierungsnase, kann das nicht passieren, denn die Sub-D-Verbindung ist verpolsicher. Hier muss das Flachbandkabel aber vor dem Zusammenbau von Pi und Modul eingesteckt werden, später ist das nicht mehr möglich.

Inbetriebnahme

Die Inbetriebnahme des PI-OBD-HATs setzt Grundkenntnisse des Raspberry Pi und des Umgangs mit Linux und der Softwareinstallation voraus. Zur Konfiguration und zur Bedienung sollte eine Tastatur und eine Maus (wenn kein Touch-Display vorhanden ist) an den Raspberry-Pi angeschlossen sein. Dann kann es losgehen. Als erste Maßnahme wird die serielle Kommunikationsschnittstelle des Pi freigeschaltet. Es würde den Rahmen dieses Kastens sprengen, dies hier zu erläutern, zumal es Unterschiede bei den verschiedenen Pi-Modellen gibt. Dier Vorgang wird aber in aller Ausführlichkeit in der Anleitung unter [10] beschrieben. Nach einem Funktionstest mit einem Terminalprogramm ist das Gerät betriebsbereit.

Der HhopenThread ist der Thread, in dem die Firmware des OBD-Analysers ausgeführt wird. Dieser Thread kommuniziert auf verschiedenste Weise mit den anderen Threads. Über das SPDR-Register werden Displaydaten zum LcdThread, über die Register OCR1A/B/C werden RGB-Werte für die LED-Hintergrundbeleuchtung zum LcdThread geschickt. Über das UDR0-Register werden AT-Kommandos und OBD-Requests zum DxmThread gesendet und Antworten empfangen. Über das WDTCR-Register wird ein Neustart der Firmware beziehungsweise von HHGui vom MainThread angefordert. Vom PINA-Register werden Tastendrücke vom MainThread empfangen. Über seine Interrupt-Service-Routinen wird der HhopenThread vom CounterThread und vom DxmThread gesteuert. Details folgen in den Beschreibungen der anderen fünf Threads.

Der MainThread ist für die Auswertung der Kommandozeilen-Parameter, die Initialisierung der seriellen Schnittstelle, das Erzeugen der anderen Threads und am Programmende auch für die Rückgabe dieser Ressourcen zuständig. Dazwischen wertet der Thread in einer Endlosschleife die Tasten aus, die vom GtkThread geliefert werden. Bei den Tasten muss zwischen den Tasten für die Steuerung des OBD2-Simulators und den Up/Down/ESC/OK-Tasten für den HhopenThread unterschieden werden. Bei den Tasten des HhopenThreads muss eine in der Firmware realisierte Tastenentprellung durchlaufen werden: Die Tasten müssen 40 ms über das PINA-Register an den HhopenThread als gedrückt gemeldet werden, damit die Firmware einen Tastendruck erkennt. Die Spanne von 40 ms wird im Zusammenspiel mit dem CounterThread gemessen. Tasten, die nur für den OBD2-Simulator benötigt werden, führen zur Änderung von Einstellungen im DxmThread.

Die Schleife zur Auswertung der gedrückten Tasten wird durch einen vom HhopenThread ausgelösten Watchdog-Reset (durch den dauerhaften Druck der ESC-Taste) oder das Beenden von HHGui durch den Druck auf die q-Taste verlassen.

Der LcdThread wertet die über die simulierte SPI-Schnittstelle vom HhopenThread empfangenen Befehle und Display-Daten aus und setzt diese in Befehle für die verwendete GTK3+-Grafikbibliothek um. Die über die OCR1A/B/C-Register gelesenen RGB-Werte für die LED-Hintergrundbeleuchtung bestimmen zusammen mit dem Kontrastwert (via SPI-Schnittstelle) die dargestellten Farben. Da das im OBD2-Analyser NG eingesetzte LC-Display einen ST7565R Display-Controller verwendet, enthält der LcdThread einen ST7565R-Simulator für die von der Firmware verwendeten ST7565R-Kommandos.

Der GtkThread wird benötigt, da die GTK3+-Hauptschleife (Funktion *gtk_main()* der GTK3+-Library) blockierend ist. GTK3+-Funktionen zum Zeichnen dürfen nur innerhalb des GtkThreads beziehungsweise im Kontext der GTK3+-Hauptschleife aufgerufen werden. Wenn der LcdThread ein Display-Update anfordert, installiert er dazu eine Callback-Funktion, die dann später von der GTK3+-Hauptschleife aufgerufen wird. Der GtkThread wertet Eingabeereignisse jeglicher Art aus und generiert daraus entweder die Up/Down/ESC/OK-Tastendrücke für den HhopenThread oder Tasten-Ereignisse, die den OBD2-Simulator steuern.

Der CounterThread dient in erster Linie dazu, den alle 10 ms auftretenden Timer2-Interrupt nachzubilden. Dazu wird alle 10 ms die Interrupt-Service-Routine (ISR) `TIMER2_COMP_vect()` aufgerufen. Außerdem hilft er dem MainThread wie beschrieben bei der Tastenentprellung.

Im OBD2-Softwaremodus arbeitet der DxmThread als Schnittstellenadapter und leitet Daten unverändert durch. Die vom UDR0-Register (USART0-I/O-Datenregister) empfangenen Daten (AT-Kommandos für das Pi-OBD-Modul oder OBD2-Requests) werden über die konfigurierte serielle Schnittstelle zum echten Pi-OBD-Modul weitergeleitet. Nach jedem gelesenen Byte wird der Firmware über den Aufruf der ISR `USART_UDRE_vect()` mitgeteilt, dass das Byte verschickt wurde und das nächste Byte in das Register geschrieben werden kann. Die Antworten vom Pi-OBD-Modul werden von der seriellen Schnittstelle empfangen und über das UDR0-Register an die Firmware weitergeleitet. Die Antwort (OK, Pi-OBD-Modul Prompt-Zeichen >, Fehler oder OBD2-Daten) wird ebenfalls byteweise ins UDR0-Register geschrieben und über den Aufruf der ISR `USART0_RX_vect()` an die Firmware gemeldet, die dann in der ISR die Daten aus dem Register liest.

Im OBD2-Simulator-Modus simuliert der DxmThread das Pi-OBD-Modul und ein oder mehrere OBD2-Steuergeräte. Dabei besteht keine Verbindung zur echten seriellen Schnittstelle. ⏮

(160204)

EINKAUFSZETTEL

→ **SKU 17944**
Modul (aufgebaut und getestet)

→ **SKU 17415**
Offizieller 7"-Touchscreen für Raspberry Pi

→ **SKU 17631**
Raspberry Pi 3

Weblinks

[1] HHEmu: www.elektormagazine.com/labs/firmware-update-and-emulator-for-obd2-analyser-ng-wireless-obd2

[2] OBD2-Analyser NG: www.elektormagazine.de/magazine/elektor-200909/3357

[3] DXM-Modul: www.diamex.de/dxshop/DIAMEX-DXM-OBD2-Modul

[4] Bluetooth-Erweiterung: www.elektormagazine.de/magazine/elektor-201004/3488

[5] HHGui: www.elektormagazine.com/labs/obd2-for-raspberry-pi

[6] www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=66368

[7] http://standards.sae.org/j1979_201408/

[8] https://en.wikipedia.org/wiki/OBD-II_PIDs

[9] www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=67245

[10] Pi-OBD-HAT: www.diamex.de/dxshop/PI-OBD-HAT-OBD2-Modul-fuer-Raspberry-PI

Elektor Labs Pipeline

Viele coole Projekte werden auf der Elektormagazine.de/Labs-Website veröffentlicht. Einige sind aufwendiger als andere, doch alle sind interessant und aus dem kostbarsten Material der Welt: ~~Liebe~~ Silizium.

Ultra-sichere Peer-to-peer-Kommunikation

Ist Ihre Kommunikation sicher? Es gibt heutzutage viele „Kryptophone“ und Krypto-Programme, aber können Sie diesen wirklich vertrauen? Trojanisierte Versionen von guten Programmen wie PGP und Software-Key-Logger für alle Arten von Betriebssystemen sind keine Ausnahme, sondern eher üblich. Bis zum Jahr 2020 werden nahezu alle mit dem Internet verbundenen Computer kompromittiert sein. Wenn Privatsphäre für Sie eine Rolle spielt, dann bauen Sie dieses Krypto-System und genießen Sie eine unzerstörbar private Peer-to-Peer-Kommunikation!

https://goo.gl/OhXNEu

Ein elektronischer Würfel, nicht in den Wolken

Manchmal wollen Sie einfach nur etwas Einfaches machen, just for fun, oder high werden von den köstlichen Dämpfen von heißem 60/40-Lot und Flussmittel. Dieser elektronische Würfel ist ein solches Projekt: einfach zu bauen, einfach zum Laufen zu bekommen und einfach zu bedienen. Brauchen Sie mehr als einen? Bauen Sie zwei! Oder drei! Es ist so einfach. Kein Mikrocontroller, kein Schaltnetzteil, keine Cloud-Verbindung, nur ein guter alter 555er und ein 4017-Johnson Zähler.

https://goo.gl/4tiAu1

Smartwatch selbstgebaut

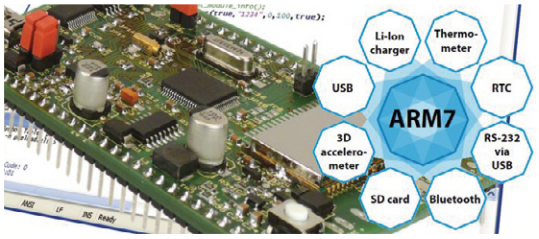
Die nWatch ist eine „Wearable“-Entwicklungsplattform in Form einer Smartwatch. Sie basiert auf einem STM32-Mikrocontroller, verfügt über einen bunten Grafik-Touchscreen, MP3-Decoder, Positionssensoren, Bluetooth 4.0 und vieles andere mehr. Der Entwurf für ein 3D-druckfähiges Uhrgehäuse ist auch vorhanden. Aber die wirklich coole Sache ist das Open-Source-Design, an dem Sie nach Herzenslust Änderungen vornehmen können. Dieses Projekt war einer der Gewinner des Fast Forward Award 2016 und wird bald veröffentlicht.

https://goo.gl/V1EUOC

The Scepter — IoT ist nichts Neues!

Das Elektor Szepter ist ein kleines ARM7 Board auf Basis des LPC2148. Das Board verfügt über Bluetooth, ein Ladegerät, eine SD-Kartenbuchse, einen Beschleunigungssensor, ein Thermometer und USB. Es wurde in Elektor 2010 als Open-Source- & Open-Hardware-Projekt veröffentlicht. Die Support-Dateien waren lange ziemlich verstreut, nachdem unsere Website erneuert, redirected, neu gehostet und was weiß ich noch wurde. Jetzt sind alle sieben Projekte und verwandte Artikel auf der Labs-Website zusammengefasst, damit Sie sie in toto genießen können.

(160293)



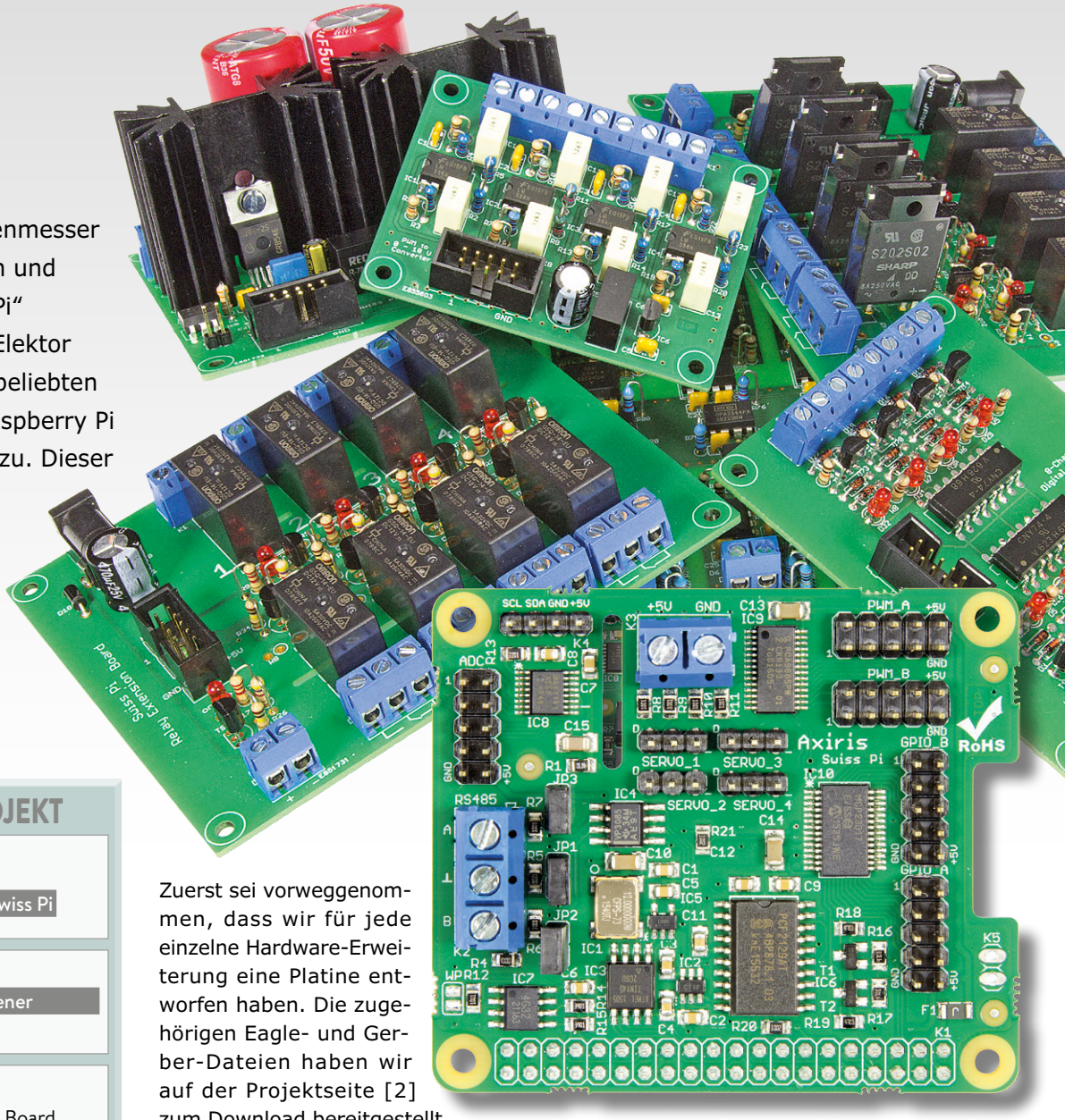
https://goo.gl/kuCiyr

Swiss Pi Extensions

Hardware-Erweiterungen

Von **Ilse Joostens** und
Peter S'heeren (B)

Unser Schweizer Taschenmesser für das Messen, Steuern und Regeln, das als „Swiss Pi" im September 2016 in Elektor erschien [1], fügt dem beliebten Einplatinencomputer Raspberry Pi wichtige Funktionen hinzu. Dieser zweite Ergänzungs-Beitrag zum Swiss Pi stellt Hardware-Erweiterungen vor.



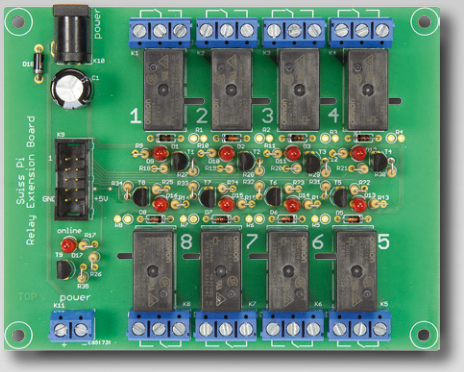
INFOS ZUM PROJEKT

- Mikrocontroller
- Raspberry Pi | Swiss Pi
- Einsteiger
- ➡ Fortgeschrittener
- Experte
- 0,5 h bis 1,5 h pro Board
- Lötkolben mit feiner Lötspitze
- 25 € bis 50 € pro Board

Zuerst sei vorweggenommen, dass wir für jede einzelne Hardware-Erweiterung eine Platine entworfen haben. Die zugehörigen Eagle- und Gerber-Dateien haben wir auf der Projektseite [2] zum Download bereitgestellt, so dass ein Hersteller beliebiger Wahl die Platinen anfertigen kann. Den Einsatz von SMD-Bauteilen haben wir möglichst vermieden. Unsere Entwürfe können auch als Richtschnur dienen, falls der Wunsch besteht, andere Hardware-Erweiterungen in eigener Regie zu entwickeln. Die hier vorgestellten Erweiterungen sind nicht auf das Swiss Pi beschränkt, sie sind ausnahmslos universell einsetzbar. Das Kombinieren mit anderen Mikrocontroller-Systemen, zum Beispiel mit dem Arduino, ist problemlos möglich.

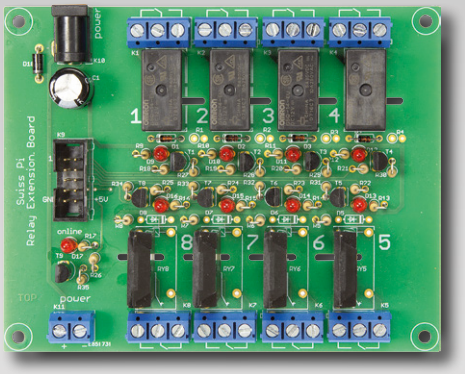
8-Kanal Relaisboard

Das 8-Kanal Relaisboard ist mit Relais der Baureihe G5Q von Omron bestückt (siehe **Bild 1**). Anstelle elektromechanischer Relais sind auch Solid-state-Ausführungen des Typs S202S02 von Sharp ein-



setzbar. Leider stellt Sharp diese Relais nicht mehr her, doch noch sind sie jedoch über eBay erhältlich. Andere elektronische Typen sind nur bedingt geeignet, denn auf die Platine passt ausschließlich der genannte Typ von Sharp. Den Eingang des S202S02 bildet eine LED ohne Strombegrenzungswiderstand, er muss extern hinzugefügt werden.

Trotz der kompakten Abmessungen erfüllt das Relais G5Q alle für Netzspannung geltenden Standards hinsichtlich der Isolierungen und Kriechstrecken. Weil die Anschlüsse des Solid-state-Relais etwas enger beieinander liegen, wurden in ihrer Nähe kleine Schlitze in die Platine eingelassen. Damit werden die vorgeschriebenen Kriechstecken eingehalten. Das Relais G5Q kann zwar Ströme bis 10 A schalten, wegen der Breite der Leiterbahnen ist der Strom jedoch auf 5 A begrenzt.



Gesteuert werden die acht Relais und die zugehörigen Betriebsanzeige-LEDs über acht Transistoren BC547. Eine Kontroll-LED leuchtet auf, wenn sowohl

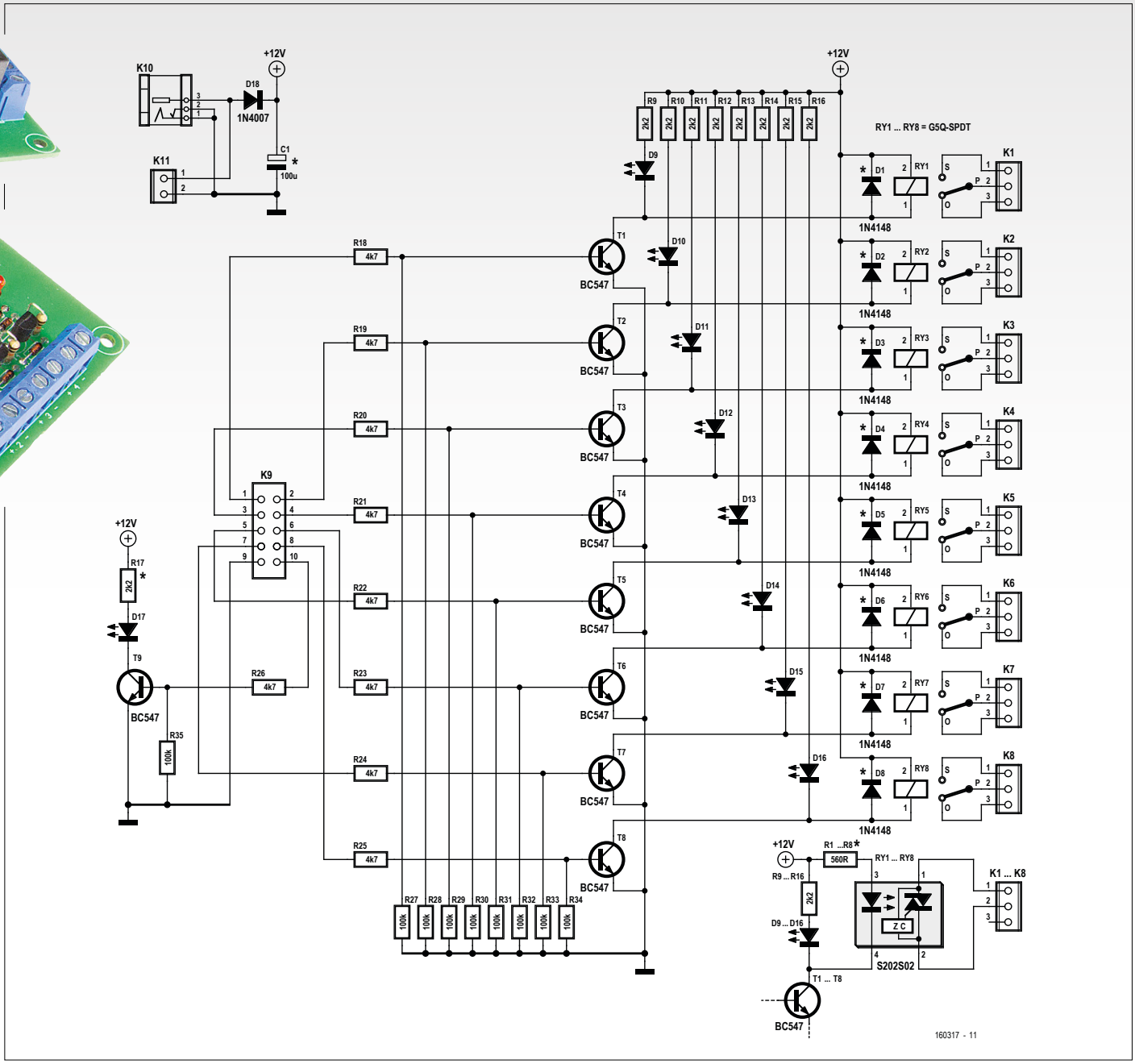


Bild 1. Auf dem 8-Kanal-Relaisboard können wahlweise elektromechanische Relais oder Solid-state-Relais montiert werden.

STÜCKLISTE RELAISBOARD

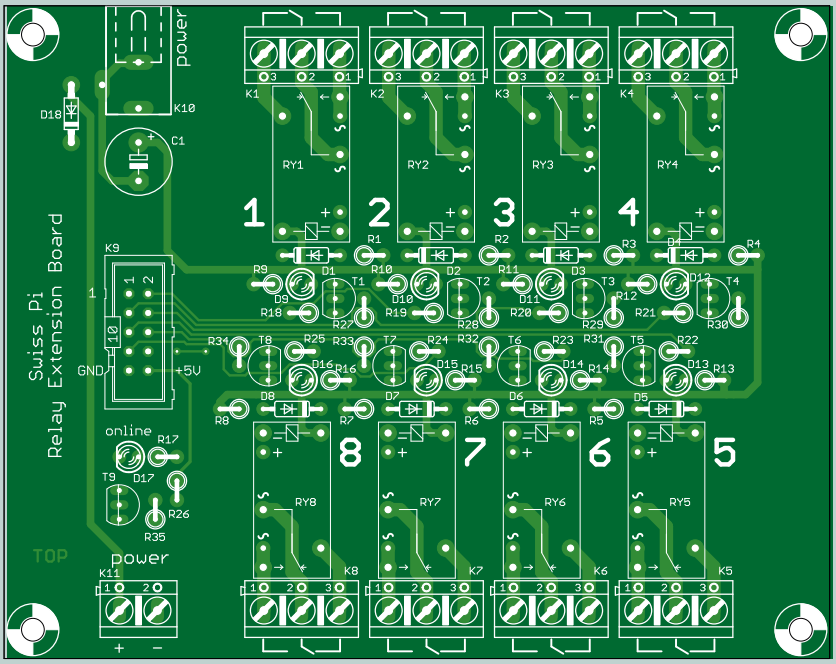
Widerstände
R1...R8 = 560 Ω*
R9...R17 = 2k2
R18...R26 = 4k7
R27...R35 = 100 k

Kondensatoren
C1 = 100 µ/16 V

Halbleiter
D1...D8 = 1N4148**
D9...D17 = LED 3 mm, rot
D18 = 1N4007
T1...T9 = BC547B

Außerdem
RY1...RY8 = G5Q-14-EU DC12V oder S202S02***
K1...K8 = Kabelschraubklemme 3-polig, RM 5 mm
K9 = Steckerleiste mit Wanne, 10-polig
K10 = Stromversorgungsbuchse mit Innenstift
K11 = Kabelschraubklemme 2-polig, RM 5 mm

* Erforderlich bei Solid-state-Relais S202S02
** Erforderlich bei elektromechanischen Relais G5Q
*** Erforderlich pro Kanal nach Wahl. Falls gewünscht, können die Relais G5Q auch an abweichender Spannung betrieben werden. In diesem Fall sind R1...R17 und eventuell auch C1 anzupassen



am Relaisboard als auch am Swiss Pi Betriebsspannung liegt.
Relais des Typs G5Q arbeiten auch an abweichenden Betriebsspannungen, zum Beispiel an 24 V. In diesem Fall müssen die Werte bestimmter Widerstände angepasst werden.

Digitales 8-Kanal-Eingangsboard

Diese Erweiterung stellt acht mit Optokopplern galvanisch getrennte Eingänge

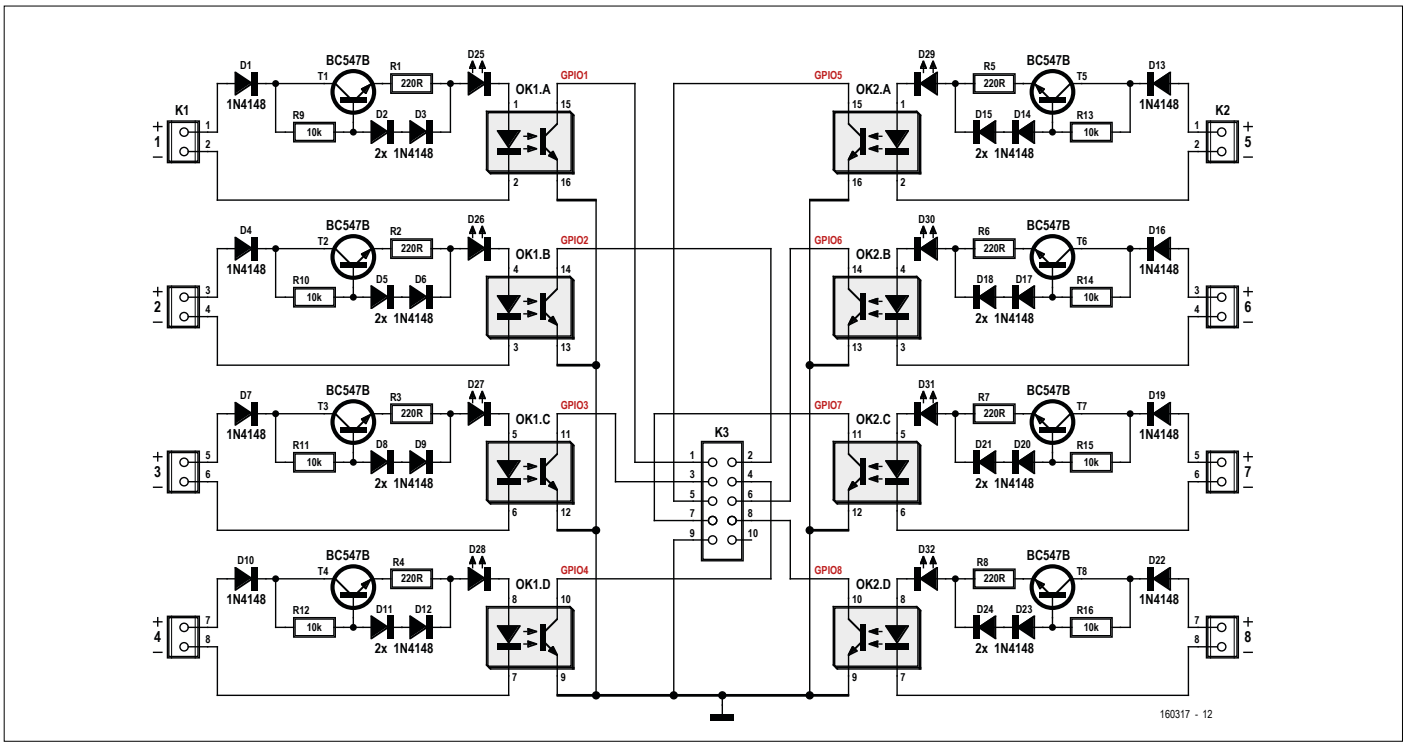


Bild 2. Optokoppler trennen die acht digitalen Eingänge galvanisch vom Swiss Pi und Raspberry Pi.

für Gleichspannungen zwischen 5 V und 45 V bereit, die Schaltung ist aus **Bild 2** ersichtlich. Auf der Eingangsseite halten Stromquellen, aufgebaut mit Transistoren BC547, die LED-Ströme im Bereich der Eingangsspannung annähernd konstant. Mit den Optokoppler-LEDs sind Betriebsanzeige-LEDs in Reihe geschaltet. Der LED-Strom berechnet sich aus den Durchlassspannungen der Dioden 1N4148 abzüglich der Basis-Emitter-Spannung des Transistors, geteilt durch 220 Ω (Widerstände R1 bis R8). Im Eingangsspannungsbereich 5...45 V bewegt sich der Strom, der durch die Dioden 1N4148 fließt, zwischen einigen zehn Mikroampere und einigen Milliampere. Aus der Durchlasscharakteristik der 1N4148 geht hervor, dass dies der Bereich ist, in dem die Durchlassspannung vom Durchlassstrom in vergleichsweise hohem Maß abhängig ist. Das bedeutet, dass der Strom nicht wirklich über den gesamten Eingangsspannungsbereich konstant ist. Konkret fließen abhängig von der Eingangsspannung etwa 3...6 mA.

Wird versehentlich die Polarität der Eingangsspannung verwechselt, liegt am Transistor eine negative Kollektor-Emitter-Spannung. Der Transistor arbeitet zwar immer noch als NPN-Typ, die Verstärkung und die Durchlassspannung

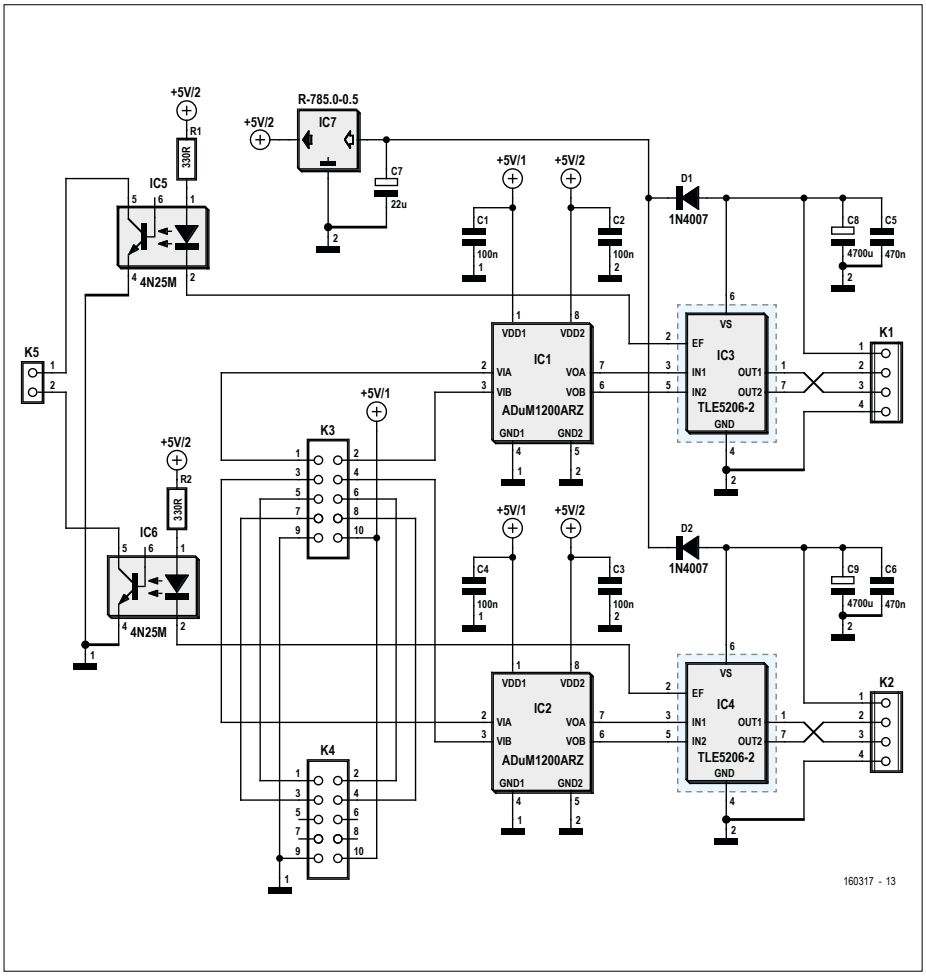


Bild 3. Die Motorsteuerungen lassen sich paarweise zusammenschalten, so dass mit dem Swiss Pi insgesamt bis zu acht DC-Motoren gesteuert werden können.

STÜCKLISTE DIGITALES EINGANGSBOARD

Widerstände

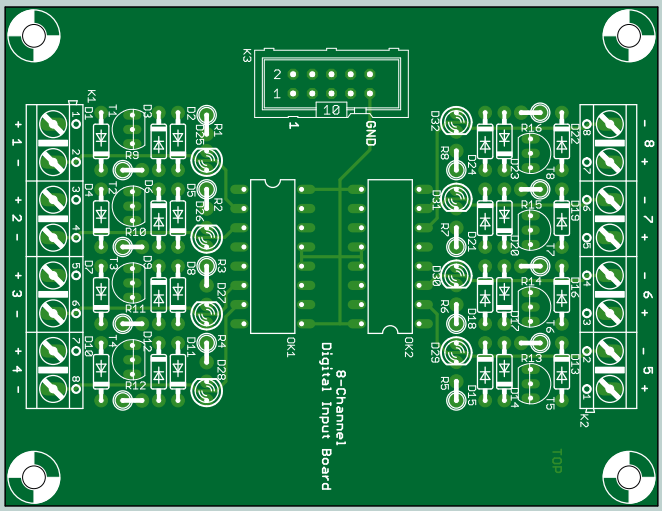
R1...R8 = 220 Ω
R9...R16 = 10 k

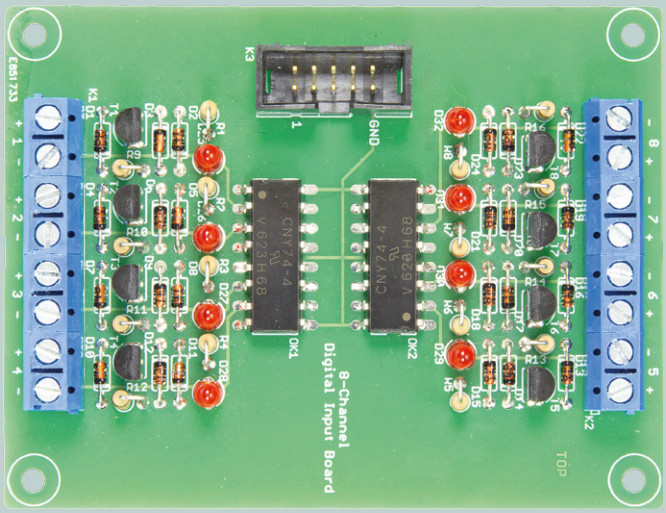
Halbleiter

D1...D24 = 1N4148
D25...D32 = LED 3 mm, rot
T1...T8 = BC547B
OK1,OK2 = CNY74-4

Außerdem

K1,K2 = Kabelschraubklemme 8-polig, RM 5 mm
K3 = Steckerleiste mit Wanne, 10-polig





sinken jedoch auf sehr niedrige Werte. Wenn die Durchlassspannung etwa -6 V unterschreitet, beginnt der Transistor zu leiten. Der Strom wird dann nur noch vom Widerstand 220 Ω sowie den Leckströmen der Betriebsanzeige-LEDs und Optokoppler-LEDs begrenzt. Um Beschädigungen der LEDs zu vermeiden, insbesondere bei hohen Eingangsspannungen, ist eine zweite 1N4148 zur Verpolsicherheit hinzugefügt.

Wenn für die Betriebsanzeige statt der roten LEDs andersfarbige LEDs verwendet werden, ist die abweichende Durchlassspannung solcher LEDs zu beachten. Der Bereich der Eingangsspannung kann sich verschieben.

Die Optokoppler-Ausgänge, die auf der Seite des Swiss Pi liegen, sind Ausgänge mit offenen Kollektoren. Deshalb müssen die GPIOs als Eingänge mit Pullup-Widerständen konfiguriert werden. Beim logischen Signal High an einem Eingang wird die zugehörige GPIO-Leitung auf Low gezogen.

DC-Motorsteuerung

Die in **Bild 3** dargestellte DC-Motorsteuerung kann zwei Gleichstrommotoren unabhängig voneinander steuern. Die Betriebsspannung kann bis 40 V betragen, die obere Grenze des Stroms liegt bei 5 A. Als Leistungskomponenten befinden sich zwei MOSFET-H-Brücken vom Typ TLE5206-2 in der Schaltung. Dieser Typ lässt sich unkompliziert steuern, er ist gegen Überlast und Kurzschluss gesichert. Da der Widerstand im Leitzustand nur 0,2 Ω beträgt, hält sich die Wärmeentwicklung in Grenzen. Nachteilig kann sich auswirken, dass die Schaltzeit (Source-on-Zeit) mit 15 µs nicht allzu kurz ist. Doch da der PWM-Controller des Swiss Pi schon bei 1,5 kHz seine Obergrenze erreicht, stellt dies kein Problem dar. Die niedrige Frequenz des steuernden Signals kann zur Folge haben, dass es über den DC-Motor hörbar ist, insbesondere wenn die PWM-Werte niedrig sind.

Welligkeiten der Betriebsspannung, die von den DC-Motoren verursacht werden, können nicht zum Swiss Pi oder Raspberry Pi vordringen. Verhindert wird dies durch die Optokoppler zwischen den H-Brücken und dem Swiss Pi. Auf der Eingangsseite werden die PWM-Signale von digitalen Isolatoren des Typs ADUM1200ARZ galvanisch getrennt. Verglichen mit Optokopplern weisen diese zweikanaligen Isolatoren deutlich höhere Bandbreiten auf. Leider ist der ADUM1200ARZ nur in SMD-Bauform erhältlich, das 8-Pin-SOIC-Gehäuse lässt sich jedoch noch manuell löten. Für die

STÜCKLISTE DC-MOTORSTEUERUNG

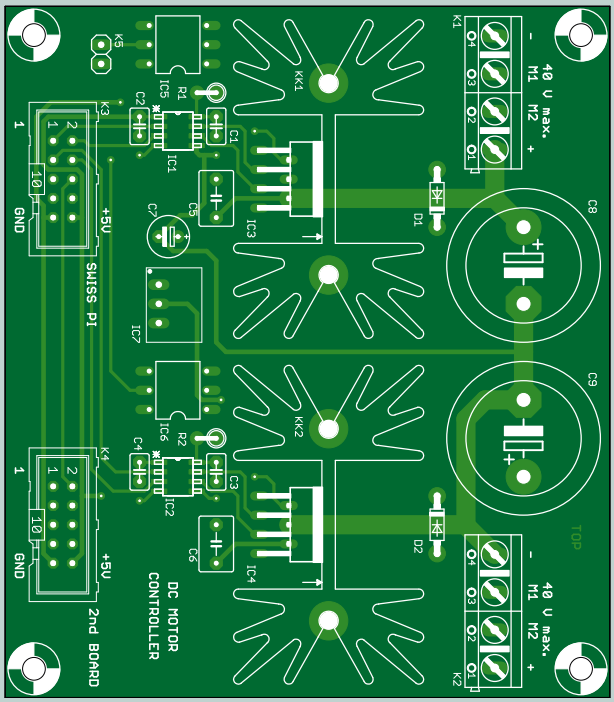
Widerstände
R1,R2 = 330 Ω

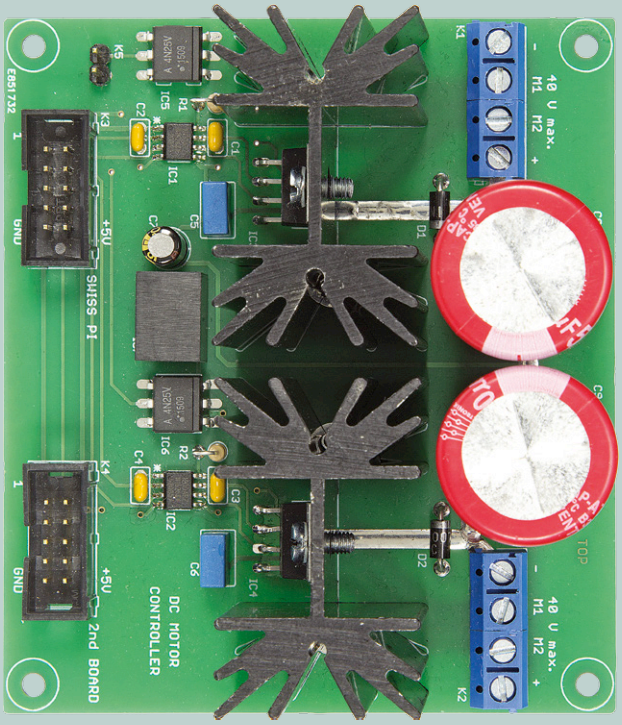
Kondensatoren
C1...C4 = 100 n
C5...C6 = 470 n/63 V*
C7 = 22 µ/50 V*
C8,C9 = 4700 µ/50 V*

Halbleiter
D1...D2 = 1N4007
IC1,IC2 = ADUM1200ARZ
IC3,IC4 = TLE5206-2
IC5,IC6 = 4N25
IC7 = DC/DC-Wandler SIP3, 5 V/1 A, Eingang 40 V (z. B. Würth 173010542)*

Außerdem
K1,K2 = Kabelschraubklemme 4-polig, RM 5 mm
K3,K4 = Steckerleiste mit Wanne, 10-polig
KK1,KK2 = Kühlkörper, z. B. SK 129 38,1 STS von Fisher Elektronik (38,1 · 42 · 25 mm)

* Falls die Motorsteuerung an Spannungen unter 40 V betrieben wird, können diese Komponenten angepasst werden. Der Strom von IC7 kann weniger als 1 A betragen, denn nur die DC/DC-Wandler mit Eingangsspannungen über 28 V liefern bis 1 A.





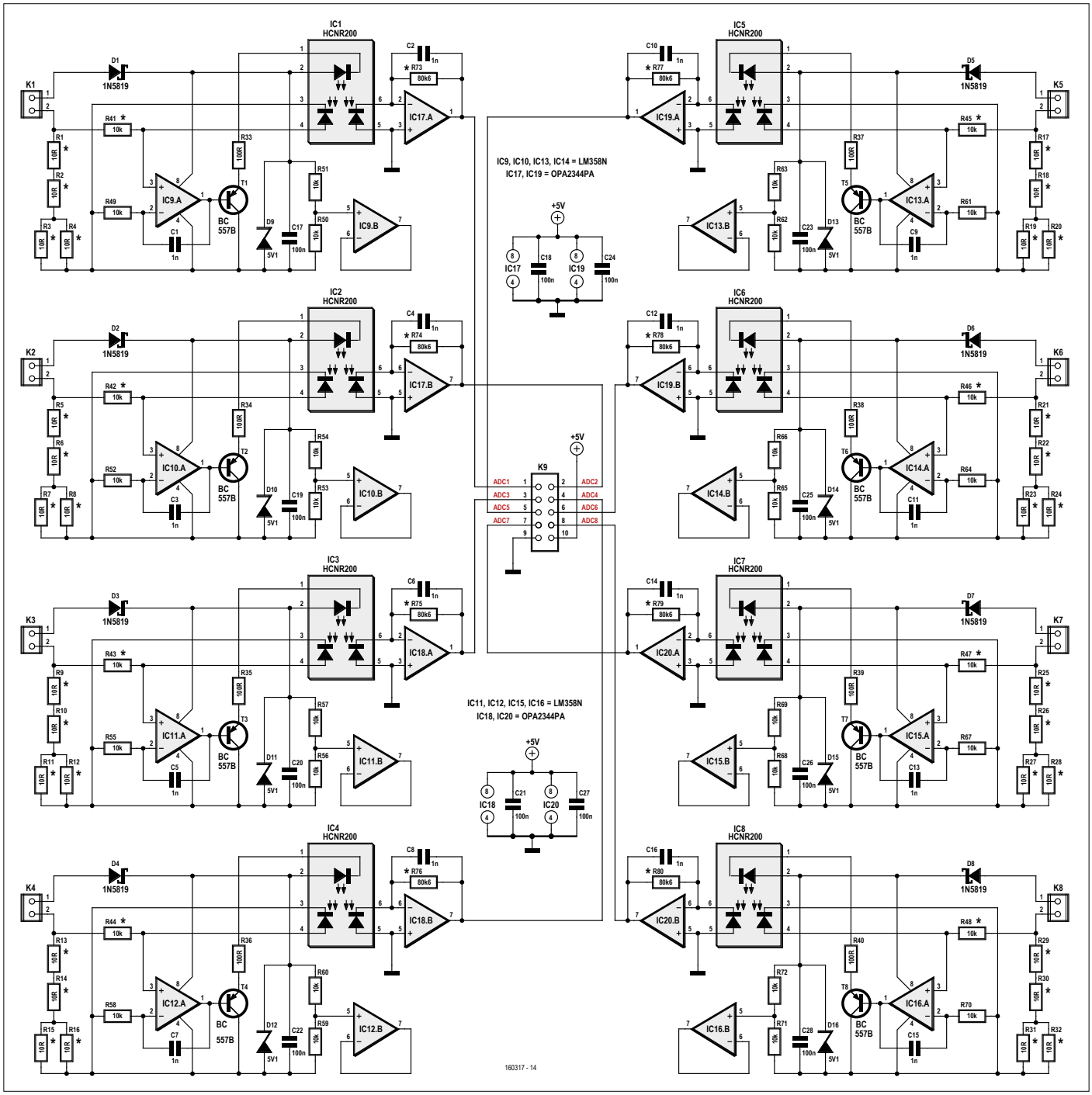


Bild 4. Das Stromschleifen-Interface erscheint ziemlich komplex, es besteht jedoch aus acht identischen Schaltungen.

optionale Rückführung der von den H-Brücken kommenden Error-Signale sind einfache Optokoppler vorgesehen.

Die Motorsteuerungen lassen sich paarweise zusammenschalten, so dass mit dem Swiss Pi insgesamt bis zu acht DC-Motoren gesteuert werden können.

Ein DC/DC-Wandler stellt die Betriebsspannung 5 V für die logischen Komponenten auf der Seite der H-Brücken bereit. Bei hohen Betriebsspannungen muss der DC/DC-Wandler Eingangsspannungen bis 40 V oder mehr vertragen.

Stromschleifen-Interface

Wenn es um das Auslesen von Sensoren geht, ist in der Industrie noch immer die „Current Loop“ beliebt. Dies ist eine ebenso unkomplizierte wie robuste Schnittstelle, die Leitungen können auch weite Distanzen überbrücken. Die Auswahl der Sensoren, die davon Gebrauch macht, ist trotz digitaler Alternativen recht umfangreich.

Die für das Swiss Pi entworfene Schnittstellen-Schaltung (**Bild 4**) arbeitet mit analogen Optokopplern des Typs HCNR200 von Avago. In diesen Optokopplern sind zwei identische Fotodioden und eine LED integriert. Auf der Eingangsseite wird eine Fotodiode benutzt, um ein Rückkopplungs-Signal auszugeben, das vom abgestrahlten Licht der LED abhängt. Die zweite Fotodiode ist der Signalempfänger. Da die Ströme durch beide Fotodioden praktisch übereinstimmen, lassen sich nahezu lineare Übertragungscharakteristiken realisieren.

Die Schaltung folgt fast unverändert dem Vorschlag aus dem Datenblatt von Avago.

STÜCKLISTE STROMSCHLEIFEN-INTERFACE

Widerstände

R1...R32 = 10 Ω*
R33...R40 = 100 Ω
R41...R48 = 10 k*
R49...R72 = 10 k
R73...R80 = 80k6*

Kondensatoren

C1...C16 = 1 n
C17...C28 = 100 n

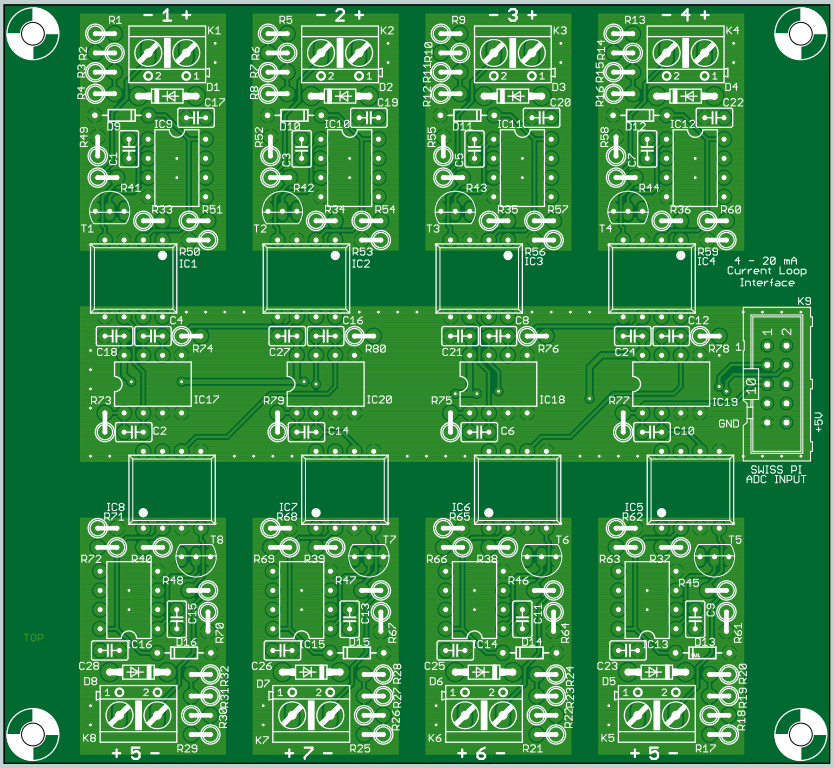
Halbleiter

D1...D8 = 1N5819
D9...D16 = Zenerdiode 5V1, 400 mW
IC1...IC8 = HCNR200
IC9...IC16 = LM358N
IC17...IC20 = OPA2344PA (oder LM6142BIN/NOPB)

Außerdem

K1...K8 = Kabelschraubklemme 2-polig, RM 5 mm
K9 = Steckerleiste mit Wanne, 10-polig

* Metallfilm 1 % oder niedriger



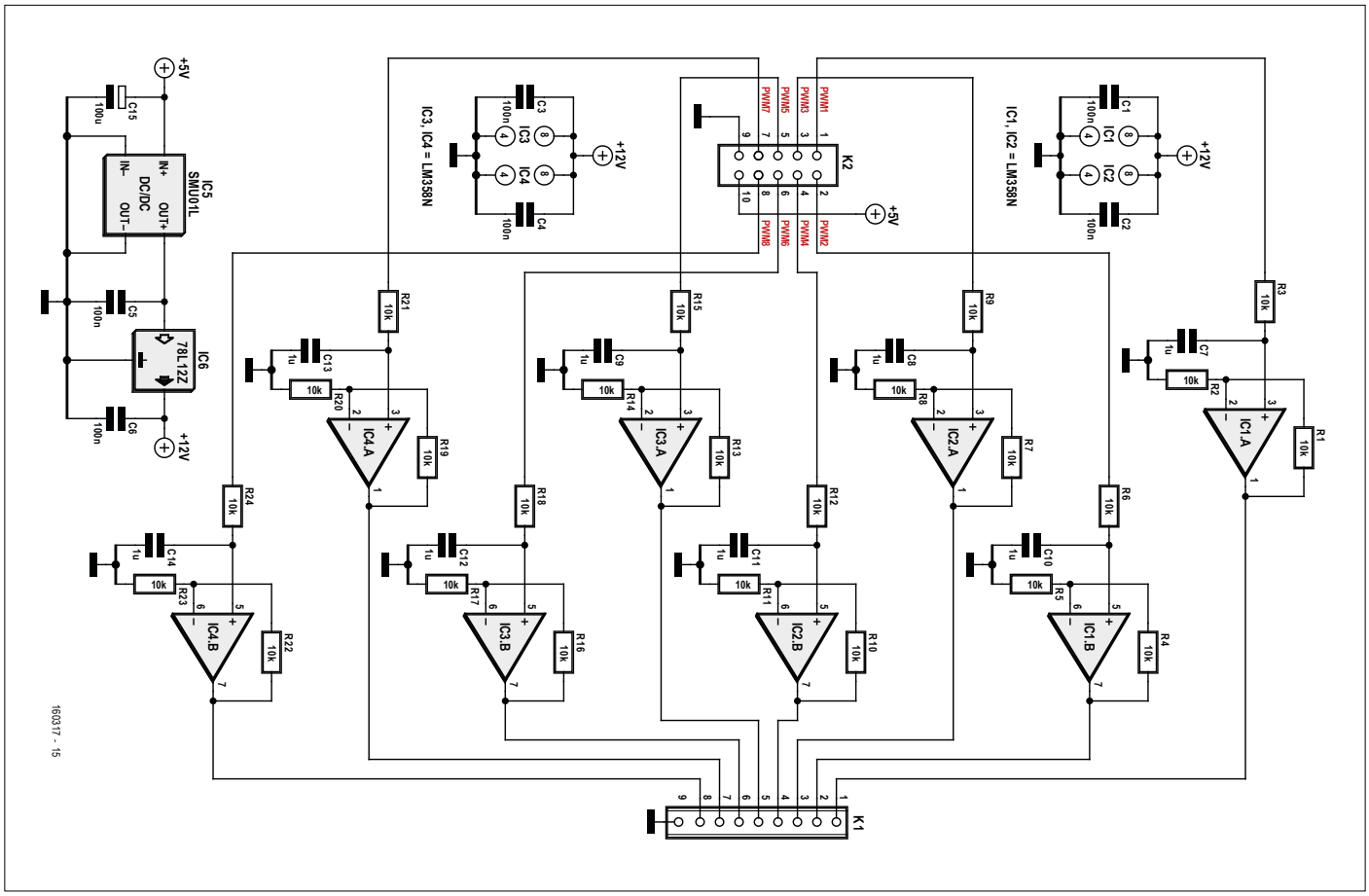
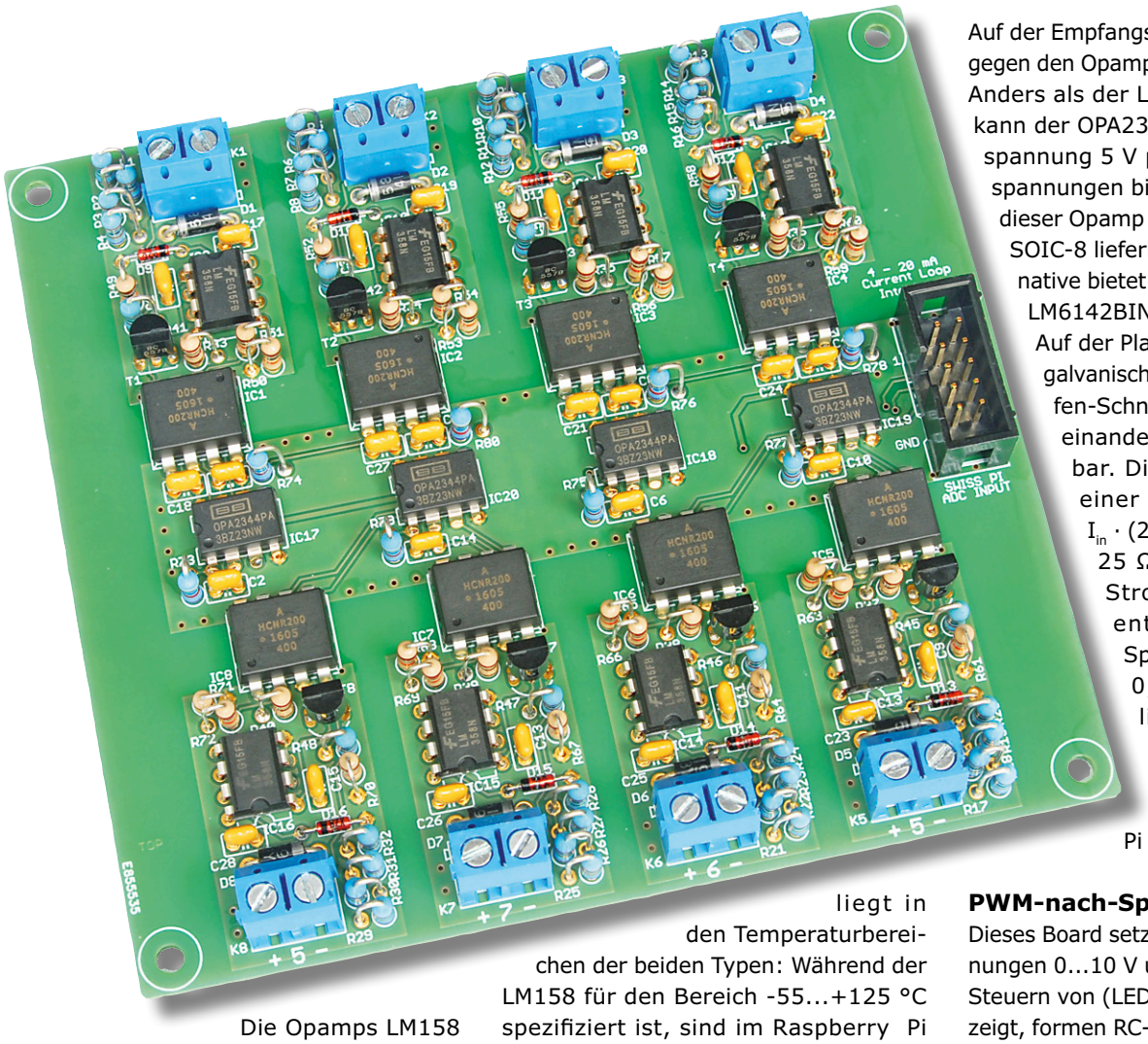


Bild 5. Der PWM-nach-Spannung-Wandler setzt PWM-Signale des Swiss Pi in Spannungen 0...10 V um.



Die Opamps LM158 auf der Sendeseite wurden gegen den Typ LM358 getauscht. Dieser Typ ist nicht nur preiswerter, sondern auch leichter beschaffbar. Der wichtigste Unterschied liegt in den Temperaturbereichen der beiden Typen: Während der LM158 für den Bereich -55...+125 °C spezifiziert ist, sind im Raspberry Pi und im Swiss Pi Bauteile für den Bereich 0...+70 °C verbaut. Der Einsatz des LM158 statt des LM358 bringt deshalb keine Vorteile.

Auf der Empfangsseite wurde der LM158 gegen den Opamp OPA2344PA getauscht. Anders als der LM158 oder der LM358 kann der OPA2344PA bei der Betriebsspannung 5 V problemlos Ausgangsspannungen bis 4 V liefern. Leider ist dieser Opamp zurzeit nur im Gehäuse SOIC-8 lieferbar. Als mögliche Alternative bietet sich unter anderem der LM6142BIN/NOPB an.

Auf der Platine befinden sich acht galvanisch getrennte Stromschleifen-Schnittstellen, sie sind voneinander unabhängig einsetzbar. Die Ausgangsspannung einer Schnittstelle beträgt $I_{in} \cdot (25\ \Omega \cdot 80{,}6\ k\Omega) / (10k + 25\ \Omega) = I_{in} \cdot 201\ \Omega$. Dem Strombereich 4...20 mA entspricht folglich der Spannungsbereich 0,804...4,02 V. Damit liegt der Spannungsbereich innerhalb des Bereichs, den der A/D-Wandler des Swiss Pi verarbeitet.

PWM-nach-Spannung-Wandler

Dieses Board setzt PWM-Signale in Spannungen 0...10 V um, beispielsweise zum Steuern von (LED-)Dimmern. Wie **Bild 5** zeigt, formen RC-Glieder Gleichspannungen aus den vom Swiss Pi kommenden PWM-Signalen. Nach dem Puffern werden die Gleichspannungen von Opamps des Typs LM358 verdoppelt.

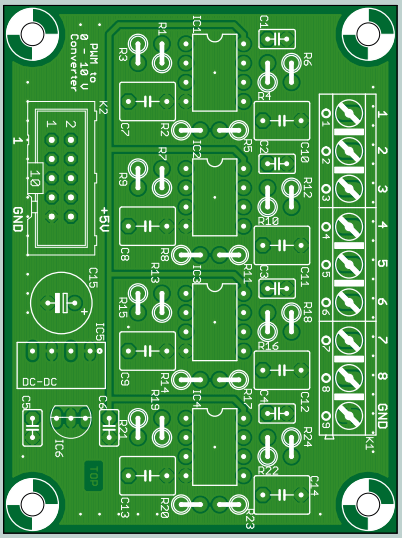
STÜCKLISTE PWM-NACH-SPANNUNG-WANDLER

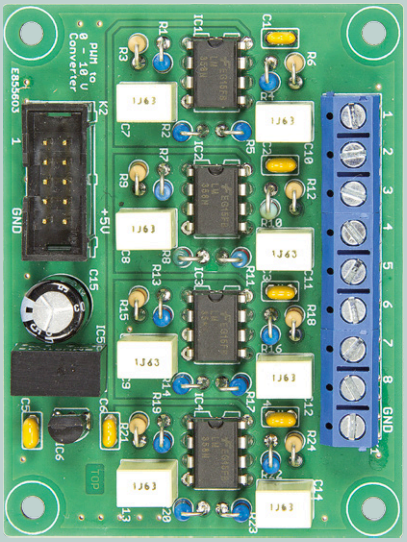
Widerstände
R1...R24 = 10 k

Kondensatoren
C1...C6 = 100 n
C7...C14 = 1 µ
C15 = 100 µ

Halbleiter
IC1...IC4 = LM358N
IC5 = SMU01L-15
IC6 = 78L12Z

Außerdem
K1 = Kabelschraubklemme 9-polig, RM 5 mm
K2 = Steckerleiste mit Wanne, 10-polig





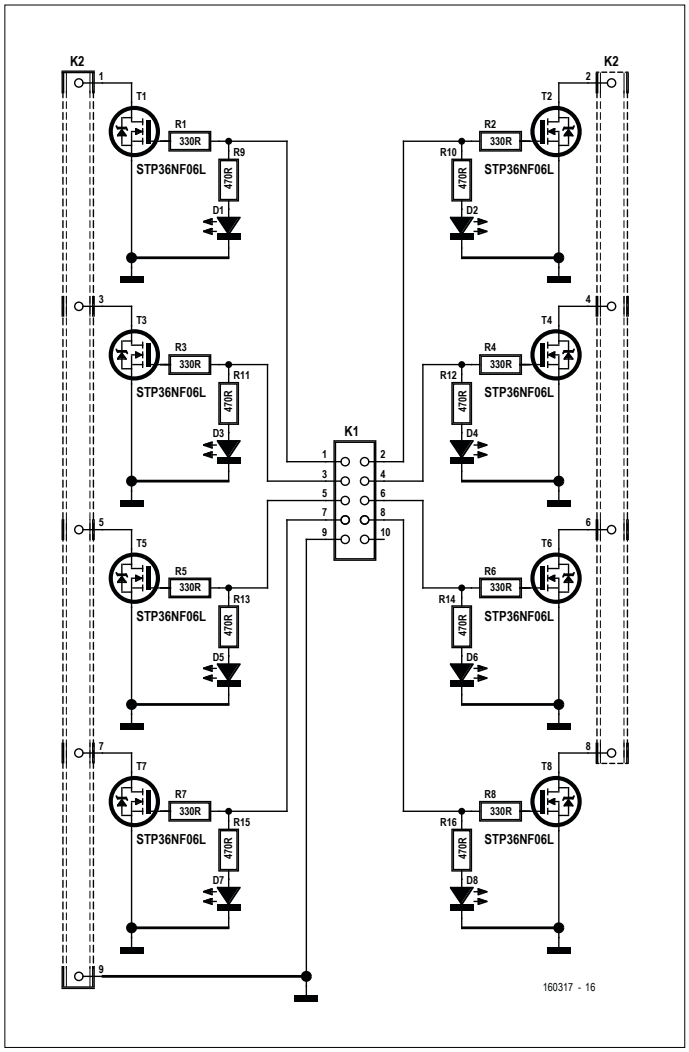


Bild 6. Auf dem MOSFET-Board kommt der MOSFET STP36NF06L zum Einsatz, er verträgt 60 V und 30 A.

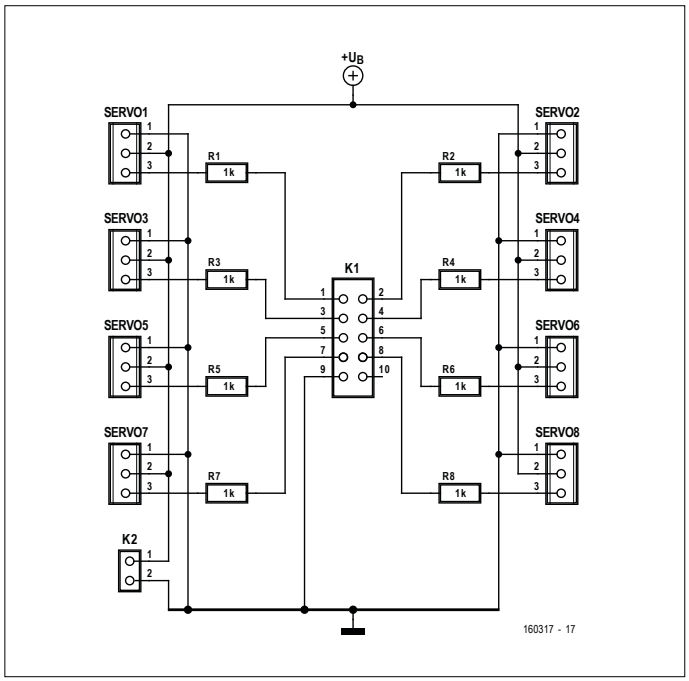


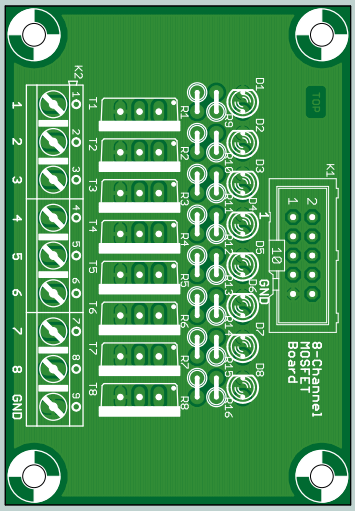
Bild 7. Über das Servo-Board kann der Raspberry Pi zusammen mit dem Swiss Pi zusätzlich acht Servos steuern.

STÜCKLISTE MOSFET-BOARD

Widerstände
R1...R8 = 330 Ω
R9...R16 = 470 Ω

Halbleiter
D1...D8 = LED 3 mm, rot
T1...T8 = STP36NF06L

Außerdem
K1 = Steckerleiste mit Wanne, 10-polig
K2 = Kabelschraubklemme 9-polig, RM 5 mm



Da die Stromaufnahme der Opamps niedrig ist und die Ausgangsspannungen 0...10 V in der Regel nur wenig belastet werden, setzt ein DC/DC-Wandler die Betriebsspannung 5 V des Swiss Pi in die Spannung 15 V um. Daraus stellt der Spannungsregler 78L12 die Betriebsspannung 12 V der Opamps bereit. In der Praxis verläuft der Bereich der Ausgangsspannungen nicht exakt von 0 V bis 10 V, sondern er beginnt bei 0 V und erreicht die zweifache Betriebsspannung des PWM-Bausteins, der sich auf dem Swiss Pi befindet.

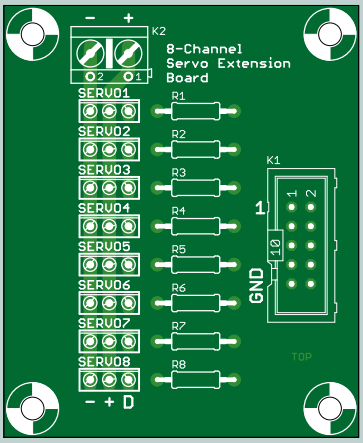
MOSFET-Board

Mit dem MOSFET-Board, dessen Schaltung aus **Bild 6** hervorgeht, werden DC-Lasten über die GPIO-Leitungen des Swiss Pi oder die PWM-Kanäle geschaltet. Ein typisches Anwendungsbeispiel für dieses Board ist das Steuern von LED-Strips. Der N-Kanal-MOSFET STP36NF06L, der hier zum Einsatz kommt,

STÜCKLISTE SERVO-ERWEITERUNGSBOARD

Widerstände
R1...R8 = 1 k

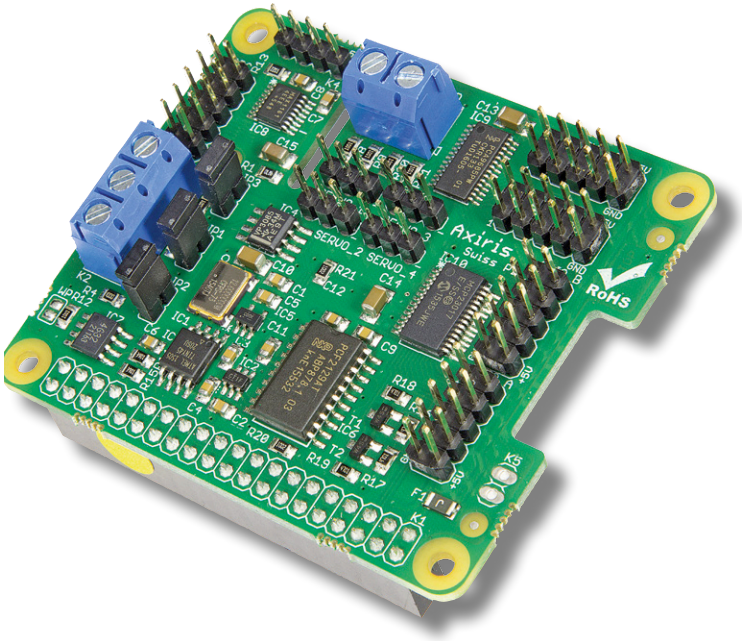
Außerdem
K1 = Steckerleiste mit Wanne, 10-polig
K2 = Kabelschraubklemme 2-polig, RM 5 mm
K3...K10 = Stiftleiste 3-polig, RM 2,54 mm



verträgt Spannungen bis 60 V bei Strömen bis 30 A. Der Durchlasswiderstand $R_{DS(on)}$ beträgt 0,032 Ω bei der Gate-Source-Spannung 10 V und dem Strom 15 A. Dieser MOSFET ist ein so genannter Logic-level-MOSFET, er schaltet auch bei der niedrigen Gate-Spannung 5 V noch zuverlässig durch. In der Praxis kann jeder Kanal ungefähr 5 A schalten, ohne dass der zugehörige STP36NF06L gekühlt werden muss. Bei allen Kanälen signalisiert eine Betriebsanzeige-LED den Schaltzustand.

Servo-Erweiterungsboard

Das Swiss Pi kann bereits vier Servos steuern. Dieses Board (Schaltung in **Bild 7**) erweitert die Anzahl der Servos; wenn zwei Exemplare miteinander kombiniert werden, bis zu maximal 16 Servos.
Die Widerstände schützen das Swiss Pi, falls der dreipolige Stecker eines Servos versehentlich in falscher Richtung aufgesteckt wird.



Wir hoffen, dass Ihnen unsere Beiträge zum Swiss Pi zusammen mit den Software-Beispielen unter [3] genügend Impulse geben, um mit dem Verbund aus Raspberry Pi und Swiss Pi das eine oder andere lohnenswerte Projekt zu realisieren. ◀

(160317)gd

Weblinks

[1] www.elektormagazine.de/150584

[2] www.elektormagazine.de/160317

[3] www.elektormagazine.de/160237

EINKAUFSZETTEL

→ SKU 17713
Swiss Pi (150584-91)

→ SKU 17631
Raspberry Pi 3

Anzeige

mouser.de

Die neuesten Produkte für Ihre neuesten Designs™

Die **größte Auswahl** der neuesten Produkte

Mehr als **4 Millionen** Produkte von über **600 Herstellern**

MOUSER ELECTRONICS

Autorisierter Distributor für Halbleiter und elektronische Komponenten

Rauschen und Empfindlichkeit bei Empfängern

Eine Einführung

Von **Robert Lacoste** (F)

Seit einigen Jahren sind neue Drahtlos-Technologien auf dem Vormarsch. Ihnen eilt der Ruf voraus, dass sie um Dimensionen besser seien als die etablierten Mitbewerber und sich gleichzeitig mit einem Zehntel an Energie begnügen. Der Datendurchsatz sei so beschleunigt worden, dass der Text eines zehnbändigen Lexikons in Sekunden auf ein Smartphone übertragbar ist (auch wenn so etwas kaum Sinn macht...). Das schien uns Anlass genug, die physikalischen Grundlagen der digitalen Funktechnik zu rekapitulieren. Mit diesem Wissen im Hintergrund fällt es leichter, tatsächliche Innovation von werbewirksamer Übertreibung zu unterscheiden.

Ein wesentlicher Faktor für die Reichweite einer Sendestation bei vorgegebener Sendeleistung ist die Empfindlichkeit des Empfängers. Doch wovon hängt die Empfindlichkeit ab? Welche Parameter sind mitbestimmend? Weshalb dauert der Download eines Videos auf ein Smartphone im ländlichen Bereich deutlich länger als im Stadtzentrum? Wenn die Antworten auf diese Fragen nicht selbstverständlich sind, kann dieser Beitrag helfen, etwas Licht in das Dunkel zu bringen. Wir versprechen, dass wir die Zusammenhänge ohne hoch wissenschaftliches Fachchinesisch beleuchten werden.

Thermisches Rauschen

Die Begriffe Empfindlichkeit und Rauschen sind eng miteinander verwoben. Wir wollen mit einem kleinen Experiment beginnen. Nehmen Sie einen Widerstand, beispielsweise 1 kΩ. Wenn Sie einen Spannungsmesser anschließen, messen Sie natürlich null Volt. Doch stimmt das wirklich? Stellen Sie das Messgerät auf Wechselspannung und messen Sie noch einmal. Vorausgesetzt das Messgerät ist genügend empfindlich, werden Sie eine Spannung größer Null messen. Dies ist die Spannung des thermischen Rauschens, die der Widerstand produziert. Das thermische Rauschen hängt von der Umgebungstemperatur ab, es wird, wie in **Bild 1** dargestellt, von der willkürlichen Bewegung der Elektronen verursacht. Thermisches Rauschen ist so genanntes *Weißes Rauschen*, was bedeutet, dass die Verteilung über den Frequenzbereich gleichförmig ist. Im Bereich 0...1 MHz hat die Rauschleistung den gleichen Betrag wie im Bereich 100...101 MHz. Die Rauschleistung, die Sie bei unserem Experiment messen, hängt von der Bandbreite (*Band Width*, BW) Ihres Spannungsmessers ab. Bei höherer Bandbreite steigt die gemessene Rauschspannung. Mit der folgenden mathematischen Beziehung lässt sich die effektive Spannung (U_{RMS}) des thermischen Rauschens an den Anschlüssen des Widerstands R berechnen:

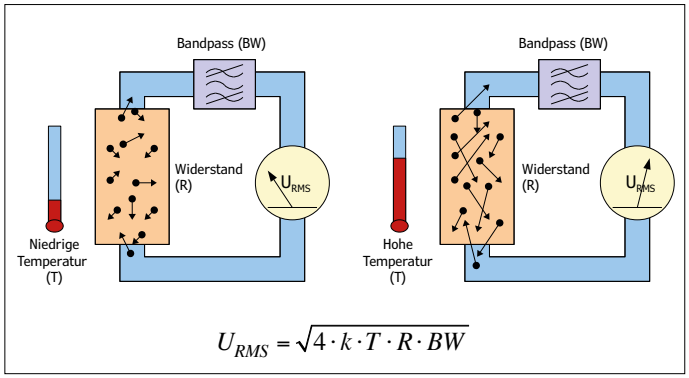


Bild 1. Jeder Widerstand produziert thermisches Rauschen.

$$U_{THERMISCH} = \sqrt{(4 \cdot k \cdot T \cdot R \cdot BW)}$$

In dieser Beziehung ist $U_{THERMISCH}$ die Spannung in Volt, T ist die Umgebungstemperatur in Kelvin, R ist der Widerstand in Ohm, BW ist die Bandbreite des Empfängers (hier der Spannungsmesser) in Hertz und k ist die Boltzmannsche Konstante, sie beträgt $1{,}38 \cdot 10^{-23}$ Joule / Kelvin. Der Widerstand 1 kΩ aus dem Experiment ergibt bei der Zimmertemperatur 20 °C und der Bandbreite 1 MHz die Rauschspannung 4 µV – nicht viel, aber messbar!

Wenn Ihnen der Begriff *Impedanz-Anpassung* nicht fremd ist, dann wissen Sie, dass für die optimale Leistungsübertragung der Lastwiderstand gleich dem Quellenwiderstand sein muss. In **Bild 2** ist deshalb der Widerstand R mit einem Lastwiderstand verbunden, der den identischen Wert R hat. Jetzt wird die Rauschspannung durch die beiden Widerstände halbiert, und die von der Last aufgenommene Rauschleistung lässt sich mit dem Ohmschen Gesetz berechnen:

$P_{THERMISCH} = U^2 / R$, wobei $U = U_{THERMISCH} / 2$ ist.

Daraus folgt:

$P_{THERMISCH} = \sqrt{(4 \cdot k \cdot T \cdot R \cdot BW)^2} / 4 \cdot R$ oder
$P_{THERMISCH} = k \cdot T \cdot BW$

Erstaunlich, dass die Rauschleistung vom Wert des Widerstands unabhängig ist, sie ist lediglich proportional zur Temperatur T und zur Bandbreite BW. Bei der Zimmertemperatur 20 °C = 294 K ist sie $P_{THERMISCH} = 4{,}05 \cdot 10^{-21}$ W pro Hertz der Bandbreite. In Dezibel ausgedrückt gilt definitionsgemäß: $P\,(dB) = 10 \log_{10} (P / P_{ref.})$. Wird die Rauschleistung in dBm ausgedrückt (Bezugsgröße 1 mW), ist $P_{THERMISCH}$ (dB) = $10 \log_{10} (4{,}05 \cdot 10^{-21}$ W / 1 mW) = −174 dBm pro Hertz. Dies ist eine wichtige Größe: Weil sich eine Empfangsantenne wie ein Widerstand verhält, beträgt der Rauschteppich des besten Empfängers der Welt bei Zimmertemperatur −174 dBm pro Hertz der Bandbreite. Als Beispiel sei der WLAN-Standard in der ursprünglichen Variante mit der Kanalbandbreite 20 MHz genannt. Das von einem WLAN-Empfänger empfangene Rauschen beträgt −174 dBm + $10 \log_{10}$ (20.000.000) dBm= −101 dBm.

Rauschfaktor

Nehmen wir an, dass ein WiFi-Empfänger ein nutzbares Signal mit der Leistung −90 dBm empfängt. Das Signal-Rausch-Verhältnis (*Signal Noise Ratio*, SNR) des von der Antenne empfangenen Signals beträgt dann −90 dBm − (−101 dBm) = 11 dBm. In jedem Empfänger ist die erste Station eines empfangenen Signals, wie **Bild 3** zeigt, eine Verstärkerstufe. Diese Stufe kann das Rauschen am Eingang nicht vom Nutzsignal unterscheiden. Das Rauschen wird deshalb in gleicher Weise wie das Nutzsignal verstärkt. Weil die Verstärkerstufe nicht ideal ist, rauscht sie ebenfalls, sie trägt mit ihrem Rauschen zum Rauschen ihres Ausgangssignals bei. Deshalb ist das Rauschen am Ausgang eines Verstärkers stets höher als am Eingang. Das Verhältnis der beiden SNR-Werte definiert den Rauschfaktor NF des Verstärkers (NF = *Noise Figure*):

$$NF = SNR_{Eingang} / SNR_{Ausgang}$$

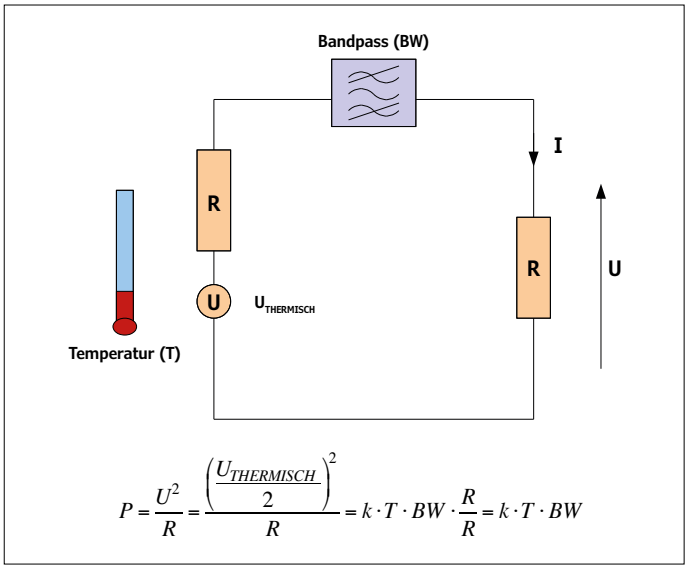
Berechnet wird der Rauschfaktor in dB (weil Logarithmen auf magische Weise Multiplikationen auf Additionen und Divisionen auf Subtraktionen reduzieren):

$$\begin{aligned} NF\,[dB] &= 10 \log_{10} (NF) \\ &= 10 \log_{10} (SNR_{Eingang} / SNR_{Ausgang}) \\ &= 10 \log_{10} (SNR_{Eingang}) - 10 \log_{10} (SNR_{Ausgang}) \\ &= SNR_{Eingang}\,[dB] - SNR_{Ausgang}\,[dB] \end{aligned}$$

Daraus geht hervor, dass der Rauschfaktor einer Verstärkerstufe gleich der Differenz in dB zwischen dem Signal-Rausch-Verhältnis am Eingang und dem Signal-Rausch-Verhältnis am Ausgang ist. Für das Beispiel des WLAN-Empfängers gilt: Wenn der Rauschfaktor der ersten Verstärkerstufe 1 dB beträgt, ist das Signal-Rausch-Verhältnis am Ausgang gleich 11 dB – 1 dB = 10 dB.

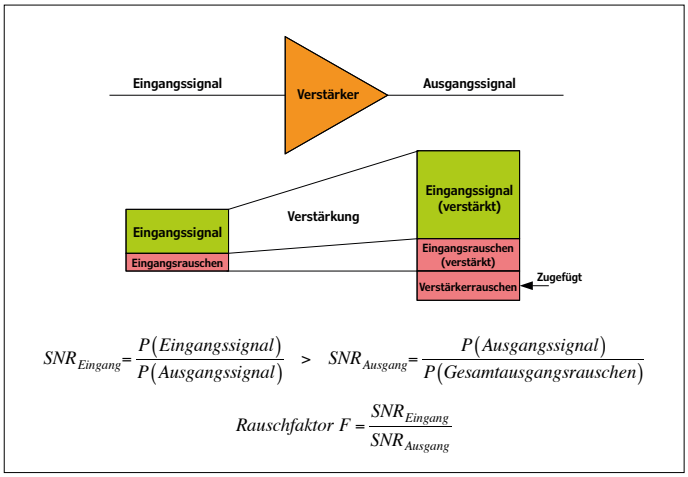
Verstärkerstufen

Zu einem realen Empfänger gehören natürlich mehrere Verstärkerstufen, das ändert jedoch nichts am Prinzip: Der Rauschfaktor des Empfängers ist das Verhältnis zwischen dem SNR am Eingang und dem SNR am Ausgang (oder die Differenz, wenn beides in dB ausgedrückt wird). Weil jede Stufe Verstärkung hinzufügt, hat das Rauschen der ersten Stufe das meiste Gewicht, denn dieses Rauschen wird von allen nachfolgenden Stufen verstärkt. Der Rauschfaktor mehrerer Stufen lässt sich mit der in **Bild 4** angegebenen Beziehung berechnen, die von



$$P = \frac{U^2}{R} = \frac{\left(\frac{U_{THERMISCH}}{2}\right)^2}{R} = k \cdot T \cdot BW \cdot \frac{R}{R} = k \cdot T \cdot BW$$

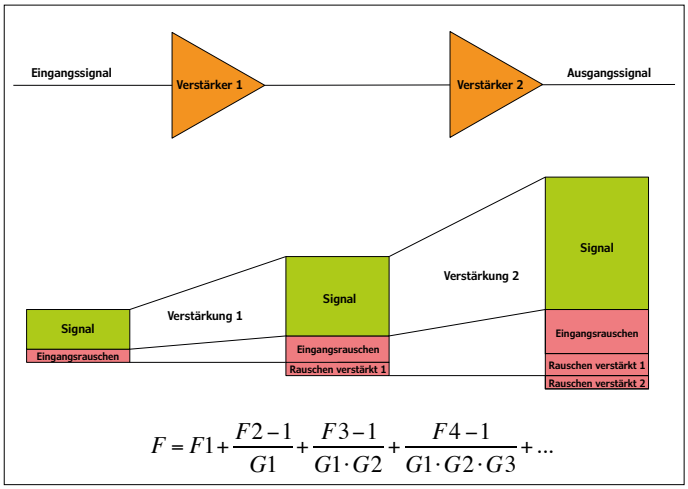
Bild 2. Die Rauschleistung beträgt k · T · BW, sie hängt von der Umgebungstemperatur und der Bandbreite des Übertragungskanals ab.



$$SNR_{Eingang} = \frac{P(Eingangssignal)}{P(Ausgangssignal)} > SNR_{Ausgang} = \frac{P(Ausgangssignal)}{P(Gesamtausgangsrauschen)}$$

$$Rauschfaktor\ F = \frac{SNR_{Eingang}}{SNR_{Ausgang}}$$

Bild 3. Der Rauschfaktor eines Verstärkers gibt die Rauschleistung an, die der Verstärker dem Eingangsrauschen hinzufügt.



$$F = F1 + \frac{F2-1}{G1} + \frac{F3-1}{G1 \cdot G2} + \frac{F4-1}{G1 \cdot G2 \cdot G3} + \ldots$$

Bild 4. Der Rauschfaktor einer Verstärkerkette lässt sich mit der von H. Friis gefundenen Beziehung berechnen. Das Rauschen der ersten Stufe hat das stärkste Gewicht.

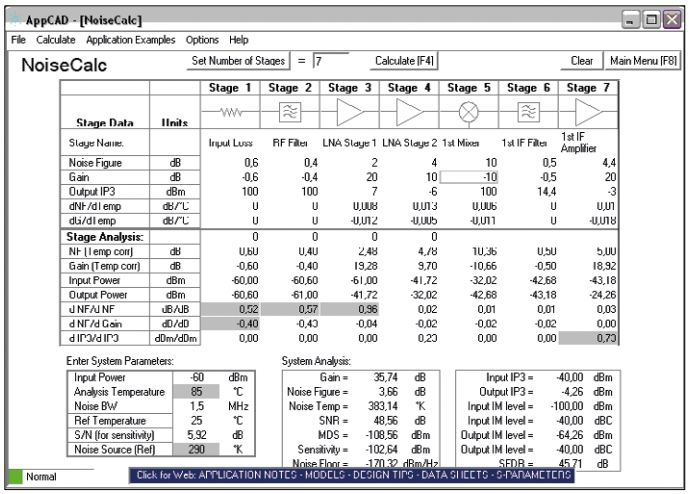


Bild 5. Ein frei verfügares Kalkulationsprogramm für Funkverbindungen: AppCad, herausgegeben von Agilent Technologies.

dem dänischen Ingenieur Harald Trap Friis (1893 bis 1976) aufgestellt wurde:

$$F = F1 + (F2-1) / G1 + (F3-1) / (G1 \cdot G2) + F4-1 / (G1 \cdot G2 \cdot G3) + \ldots$$

In dieser Beziehung sind F1, F2, F3, ... die Rauschfaktoren der einzelnen Stufen, und G1, G2, G3, ... sind die zugehörigen Verstärkungen. Allerdings wird hier mit linearen Rauschfaktoren gerechnet (nicht in dB), was die Rechenarbeit verkompliziert. Deshalb sind Programme in Form von Spreadsheets oder Apps hilfreich, die das Berechnen der Parameter von Verstärkerketten als Funktion der einzelnen Elemente übernehmen. Ein unkompliziertes Beispiel ist das Programm AppCad von Agilent (**Bild 5**), das frei aus dem Netz heruntergeladen werden kann. Mit den nötigen Messgeräten lässt sich der Rauschfaktor eines Verstärkers schnell ermitteln. Vorausgesetzt das Rauschen übersteigt einen bestimmten Mindestwert, kann ein Spektrumanalyser den Rauschfaktor überschlägig als Zahlenwert anzeigen. Mit einem Spektrumanalyser lassen sich auch präzisere Verfahren durchführen, allerdings wird dazu eine separate Rauschquelle benötigt. Näheres zu diesem Thema enthält die Application Note 57-2 von Agilent Technologies, auch sie ist im Netz verfügbar. Darüber hinaus sind aufwendige spezialisierte

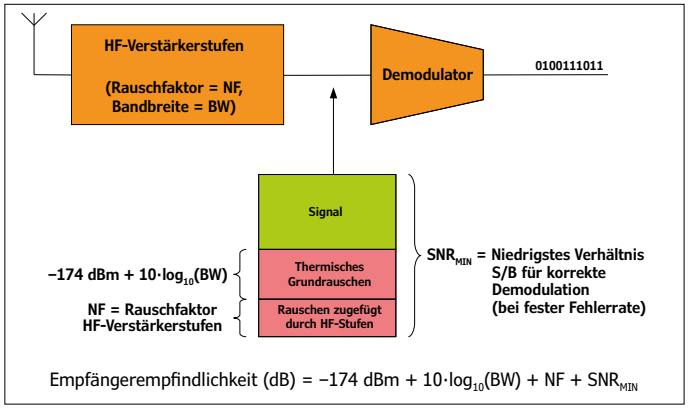


Bild 6. Das Berechnen der Empfindlichkeit eines Empfängers ist nicht schwierig...

Messgeräte auf dem Markt, wie zum Beispiel der altbewährte *Noise Figure Analyzer* HP8970B von Hewlett Packard. Dieses Gerät wendet das zweite genannte Verfahren an, wobei die Messabläufe automatisiert sind.

Vom Rauschen zur Empfindlichkeit

Nachdem wir das Rauschen behandelt haben, kommen wir zur Kernfrage: Wovon hängt die Empfindlichkeit eines Empfängers ab? Kurz zusammengefasst gehören zu einem Empfänger eine Antenne, mehrere Verstärker- und Filterstufen sowie ein Demodulator, dargestellt in **Bild 6**. Der Demodulator stellt fest, ob ein empfangenes Bit 0 oder 1 ist. Die Empfindlichkeit eines Empfängers ist als die Antennenleistung definiert, bei der die korrekte Unterscheidung von 0 und 1 gerade noch möglich ist. Dabei darf die Fehlerrate (*Bit Error Rate*, BER) die festgelegte erlaubte Toleranz nicht übersteigen.

Es dürfte nicht schwierig zu erraten sein, dass die Fehlerrate niedrig liegt, wenn das Signal-Rausch-Verhältnis am Eingang hoch ist. Oder anders ausgedrückt: Unter dem Einfluss des Rauschens wird es für den Demodulator mühsam, die Unterschiede zwischen 0 und 1 zu erkennen. Die Empfindlichkeitsschwelle am Eingang des Demodulators wird bei einem Mindestwert SNR_{MIN} unterschritten. Der Wert hängt nicht nur von der festgelegten Fehlertoleranz ab, sondern auch vom internen Aufbau des Demodulators sowie vom angewendeten Modulationsverfahren. Lassen wir den vorstehenden Absatz einen Moment lang wirken: Die Schwelle der Empfindlichkeit eines Empfängers ist erreicht, wenn das SNR am Demodulator-Eingang gleich SNR_{MIN} ist. Dieses SNR ist die Differenz des Leistungspegels, mit dem das Signal empfangen wird (definitionsgemäß die Empfindlichkeit), und dem Rauschen am Demodulator-Eingang. Letzteres setzt sich aus dem von der Antenne aufgenommenen Rauschen (-174 dBm + 10 $\log_{10}$(BW)) und dem Rauschen der Verstärkerkette im Empfänger zusammen. Das aber ist definitionsgemäß gleich dem Rauschfaktor NF. Es gilt also:

$$SNR_{min} = \text{Empfindlichkeit} - (-174 + 10\log_{10}(BW) + NF)$$

Daraus folgt:

$$\text{Empfindlichkeit [dB]} = -174\ \text{dBm} + 10\ \log_{10}(BW) + NF + SNR_{min}$$

Diese Beziehung sagt aus, dass die Empfindlichkeit jedes Empfängers ausschließlich von der Bandbreite des Empfangskanals (BW), vom Rauschfaktor (Güte der Elektronik) und vom erlaubten mindesten SNR abhängt, bei dem das korrekte Dekodieren der empfangenen Bits noch möglich ist.

Bandbreite und Übertragungsgeschwindigkeit

Bisher haben wir nur die Kanalbandbreite (BW) ins Kalkül gezogen, nicht aber eine andere fundamentale Größe: Die genutzte Übertragungsgeschwindigkeit, Kurzzeichen B (Anzahl der übertragenen Bits in der Sekunde). Über einen Kanal mit 1 MHz Bandbreite lässt sich ein Bitstrom mit 100 bps viel bequemer und mit niedriger Fehlerrate übertragen als der Bitstrom eines Videos mit 10 Mbps. Wie die Mathematik zeigt, ist das Verhältnis B/BW entscheidend, also die Anzahl der zu transportierenden Bits in der Sekunde, geteilt durch die Kanalbandbreite in Hertz. Mit anderen Worten: Mit identischen Demodulatoren besteht kein Unterschied, ob ein Bitstrom von 1 bps in einem

1 Hz breiten Kanal dekodiert werden muss oder ein Bitstrom von 1 Mbps in einem 1 MHz breiten Kanal.
Deshalb ist es gerechtfertigt, davon auszugehen, dass SNR_{min} gleich dem Produkt aus B/BW und einer Konstanten ist. Eine Charakteristik jedes Modulationstyps ist das Diagramm, das die theoretische Fehlerrate als Funktion des Verhältnisses Eb/N0 für den idealen Demodulator darstellt. Das Verhältnis Eb zu N0 (*energy per bit to noise power spectral density ratio*) ist gewissermaßen das Signal-Rausch-Verhältnis bezogen auf 1 bit, wobei Eb die Energie pro Bit und N0 die Spektraldichte des Rauschens sind. Beispielsweise lässt die GMSK-Modulation (angewendet in 2G-Netzwerken) ein viel niedrigeres Verhältnis Eb/N0 zu als die FSK-Modulation (angewendet zum Beispiel bei Garagentor-Fernsteuerungen). Auch mit fehlerkorrigierenden Codierungen lässt sich das Verhältnis Eb/N0 verbessern. Als Beispiel gibt **Bild 7** das Verhältnis Eb/N0 für die klassische FSK-Modulation wieder. Wenn die erlaubte Fehlerrate (BER) beispielsweise auf 1 bit / 1000 bit (= 10^{-3}) festgelegt wurde, ist Eb/N0 ablesbar. In dem Fall liegt sie bei etwa 11 dB.
Unter der Annahme, dass das Verhältnis B/BW den Ausschlag gibt und der Multiplikator gleich Eb/N0 ist, gilt die Beziehung:

$$SNR_{min} = Eb / N0 \cdot B / BW$$

oder in Dezibel:

$$SNR_{min}\,[dB] = Eb / N0\,(dB) + 10 \cdot \log_{10}(B) - 10 \cdot \log_{10}(BW)$$

Nun wird es spannend: Was geschieht, wenn dieser Ausdruck für SNR_{min} in die Beziehung eingesetzt wird, die für die Empfindlichkeit eines Empfängers gilt?

$$\text{Empfindlichkeit}\,[dB] = -174\,dBm + 10 \cdot \log_{10}(BW) + NF_{Kette} + SNR_{min}$$

Daraus ergibt sich:

$$\text{Empfindlichkeit}\,[dB] = -174\,dBm + 10 \cdot \log_{10}(BW) + NF_{Kette} + Eb / N0\,(dB) + 10 \cdot \log_{10}(B) - 10 \cdot \log_{10}(BW)$$

Die Terme mit BW heben sich auf:

$$\text{Empfindlichkeit}\,[dB] = -174\,dBm + NF_{Kette} + Eb / N0\,(dB) + 10 \cdot \log_{10}(B)$$

Betrachten wir diese Beziehung, dann stellen wir fest: Die Empfindlichkeit eines Empfängers hängt von nur drei Faktoren ab:

Die Güte der Elektronik (über den Rauschfaktor NF),
die genutzte Übertragungsgeschwindigkeit (B),
die Konstante Eb/N0, eine inhärente Eigenschaft des Demodulators.

Das ist alles! Die Kanalbandbreite ist nicht von Bedeutung, ebenso wenig wie die Verstärkung der Verstärkerstufen.
Jetzt ist klar, dass eine Funkverbindung eine wesentlich höhere Reichweite hat, wenn die Übertragungsgeschwindigkeit niedrig ist. Deshalb reduzieren Mobiltelefone und WLAN-Adapter ihre Übertragungsgeschwindigkeiten, wenn die Distanz zur Basisstation zunimmt. Diese Systeme sind so „intelligent", dass sie bei sinkender Feldstärke nicht die Verbindung abbrechen, sondern die Geschwindigkeit reduzieren. Das ist auch der Grund, weshalb die ersten Raumsonden nur wenige Bit in der Minute zur Erde übertrugen und Niedriggeschwindigkeitstechniken wie LoRa oder Sigfox extreme Distanzen bei nur geringem Energiebedarf überbrücken.

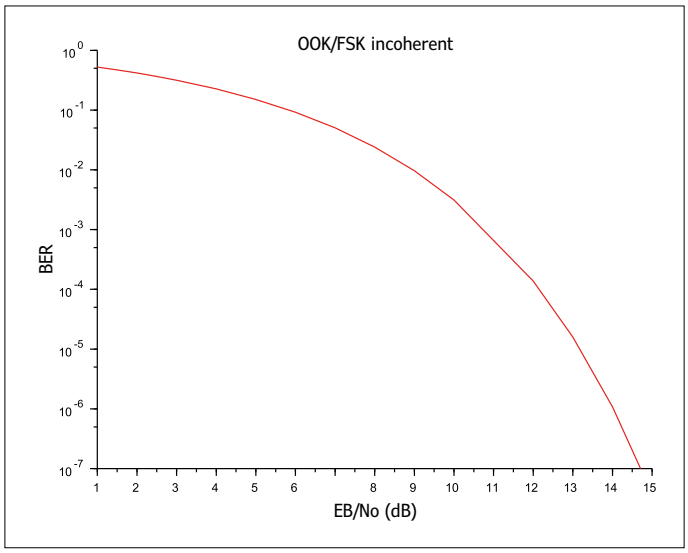


Bild 7. Der Quotient Eb/N0 als Funktion der zulässigen Fehlerrate bei der klassischen FSK-Modulation.

Schlussbetrachtung

Die Empfindlichkeit eines Empfängers ist keine magische Größe, sie lässt sich unkompliziert aus gegebenen Größen berechnen. Wenn Sie noch nicht überzeugt sind, schauen Sie sich die technischen Daten eines willkürlichen funktechnischen Geräts an (beispielsweise eines WLAN-Adapters). Dort werden Sie feststellen, dass für die Reichweite niedrige Werte bei hohen Übertragungsgeschwindigkeiten angegeben sind, und natürlich umgekehrt.
In einem folgenden Beitrag wollen wir ein anderes System betrachten, das mit niedrigen Geschwindigkeiten arbeitet, aber hohe Reichweiten erzielt: Es heißt LoRa. ⏮

(160307)gd

Dieser Beitrag ist auch in Circuit Cellar Nr. 249 erschienen.

Weblinks und Quellen

[1] Software AppCad (Agilent Technologies): www.hp.woodshot.com

[2] Noise reduction techniques in electronic systems, Henry W. Ott, John Wiley & Sons, ISBN 0-471-85068-3

[3] Short-range Wireless Communication, Alan Bensky, Newnes/Elsevier, ISBN 0-7506-7782-1

[4] Johnson-nyquist noise, Wikipedia, http://en.wikipedia.org/wiki/Johnson%E2%80%93Nyquist_noise

[5] Noise Figure Measurement Accuracy – The Y-Factor method, Application Note 57-2, Agilent technologies

[6] Spectrum analysis – Noise figure measurement, Application Note 150-9, Hewlett Packard

[7] Three methods of Noise figure measurement, Application Note 2875, Maxim

HOMELAB PROJECT

Theremin im Eigenbau

Mit JFETs statt Röhren

Das Theremin ist eine schon fast 100 Jahre alte Erfindung, die immer noch begeistert. Ein berührungslos gespieltes Musikinstrument, das jeder schon mal gehört hat, erfunden 1920 von Lew Termen alias Leon Theremin. Damals lief das Gerät mit Röhren, aber mit JFETs geht es auch. Hier wird die Schaltungsentwicklung für ein experimentelles Eigenbau-Theremin vorgestellt.

Von **Burkhard Kainka** (D)

Das Prinzip ist einfach und beruht auf der Überlagerung zweier HF-Signale, bei der ein Schwebungston entsteht. Ein HF-Oszillator wird über die Handkapazität abgestimmt und verändert die Tonhöhe (Pitch-Oszillator), der zweite Oszillator arbeitet auf einer festen Frequenz. Beide gelangen auf einen Demodulator, der ähnlich wie in einem AM-Radio aufgebaut ist. Das alles funktioniert wie beim Empfang unmodulierter Morsesignale. Und deshalb gelingt der erste Versuch auch ohne viel Aufwand mit einem vorhandenen Empfänger.

Erste Experimente mit dem Elektor SDR-Shield

Für die ersten Versuche muss man einfach nur einen Oszillator aufbauen, der irgendwie durch die Handkapazität ausreichend verstimmt werden kann. Dazu wurde das Elektor SDR-Shield auf einem Arduino Uno mit der Software für einen Standalone-VFO und der LCD-Anzeige auf dem Elektor Extension-Shield verwendet. Diese Anordnung wurde bereits in Elektor für den Einsatz im Kurzwellenbereich vorgestellt [1]. Das Shield wird hier wie ein Direktmischer verwendet und direkt mit einem aktiven Lautsprecher verbunden. Es ersetzt damit den zweiten Oszillator und den Mischer im originalen Theremin. Man kann die Überlagerungsfrequenz bequem und präzise einstellen (**Bild 1**) und braucht erstmal nur den Pitch-Oszillator. In dieser Form hat man bereits ein erstes einfaches Theremin, allerdings noch ohne Volume Control. Eine interessante Anwendung für das SDR-Shield.

Der FET-Oszillator nach **Bild 2** arbeitet bei 420 kHz und wird durch Annäherung mit der Hand um bis zu 5 kHz verstimmt. Die SDR-Antenne ist ein Stück Draht, das einfach nur nahe beim Oszillator liegt. Das Signal ist so kräftig, dass man nur eine sehr lose Kopplung braucht. Am Stereo-Ausgang wird diesmal kein PC angeschlossen, sondern direkt ein aktiver Lautsprecher. Das SDR-Shield arbei-

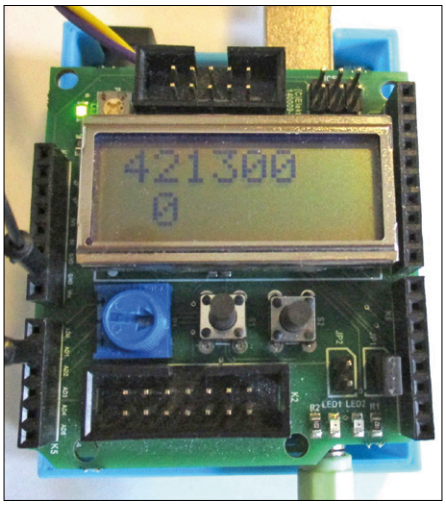


Bild 1. Der Empfänger, abgestimmt auf 421,3 kHz.

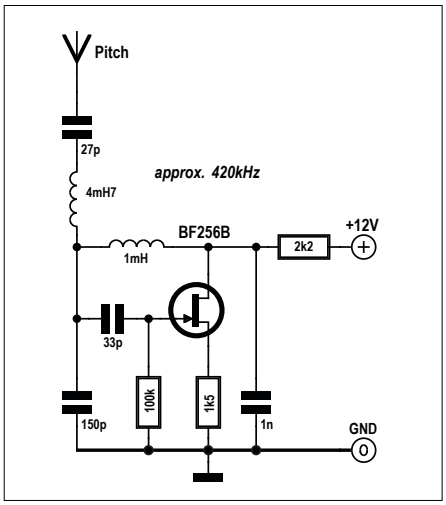


Bild 2. Ein einfacher Pitch-Oszillator.

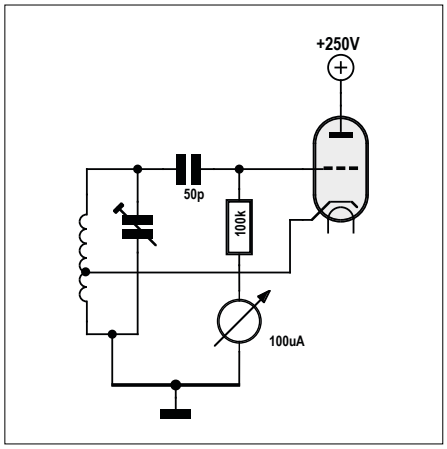


Bild 3. Ein Grid-Dip-Meter mit einer Triode.

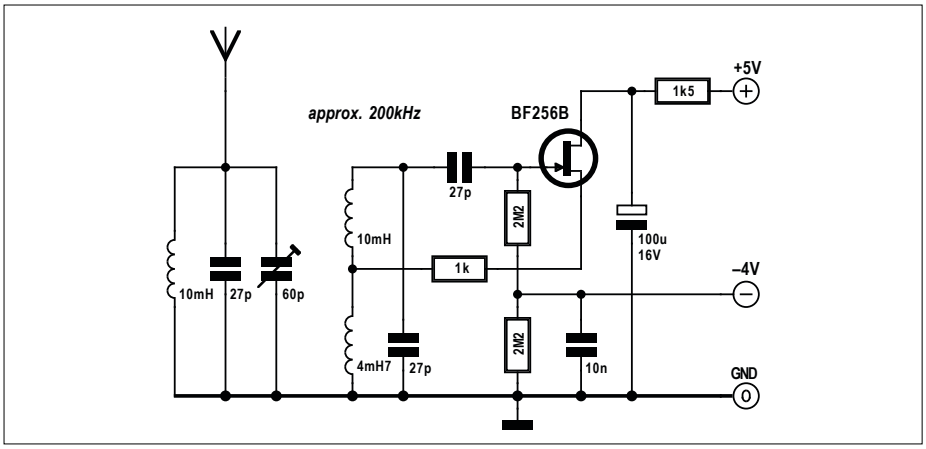


Bild 4. Angepasstes FET-Dip-Meter.

tet also ohne SDR-Software als einfacher Direktmischer.

Mit der Standalone-Software für das SDR-Shield und dem Extension-Shield erreicht man eine bequeme und genaue Abstimmung. Man sucht zunächst mit den Tasten auf dem Extension-Shield das Oszillatorsignal und stimmt es dann auf Schwebungsnull ab. Bei Annäherung mit der Hand wird die Frequenz des Pitch-Oszillators kleiner und der Schwebungston höher. Und nun ist es nur noch eine Frage der Übung, bis man damit eine erste Melodie spielen kann.

In der Antennenleitung gibt es eine Verlängerungsspule von 4,7 mH. Sie bewirkt, dass die sehr geringe Handkapazität von etwa einem Pikofarad eine ausreichend große Frequenzänderung bringt. Die Spule mit 1 mH benötigt eine Kreiskapazität von 143 pF, damit sie auf 420 kHz in Resonanz kommt. Mit 144 pF landet man gerade einmal 1,5 kHz tiefer. Die Verlängerungsspule macht aus dem einen Pikofarad wesentlich mehr. Man kann es sich bildlich so vorstellen, dass eine sehr viel längere Antenne neben einer sehr viel größeren Hand steht. Der Trick mit der Verlängerungsspule wurde auch schon beim ursprünglichen Theremin angewandt und immer wieder aufgegriffen.

Etwas wissenschaftlicher betrachtet hat ein Serien-Resonanzkreis nahe seiner Resonanz einen geringen Blindwiderstand, der unterhalb der Resonanz einer großen Kapazität entspricht. Für die verwendete Spule wurde zusammen mit einer üblichen Stabantenne eine Eigenresonanz bei 600 kHz gemessen. Das liegt relativ nahe an der Arbeitsfrequenz von 420 kHz und führt zu einer wirksamen Vergrößerung der Handkapazität. Insgesamt ist die Empfindlichkeit gegenüber der Handkapazität sehr gut, so dass man einen angenehmen Spielbereich von rund 30 cm hat. Die Frequenzstabilität ist wegen der geringen Oszillatorfrequenz voll ausreichend und mit dem SDR-Shield ergibt sich bei schwacher Kopplung ein angenehmer Klang.

Versuche zur Volume-Antenne

Einen Schwachpunkt hat dieser einfache Theremin-Oszillator aber noch. Man kann nämlich die Lautstärke nicht verändern. Das originale Theremin hat dagegen eine zweite Antenne, die man mit der linken Hand bedient. Bei einer Annäherung wird der Ton leiser und verschwindet sogar ganz. Erst damit ist das Instrument richtig zu spielen.

Der erste Gedanke ist vielleicht, dass man eines der beiden Oszillatorsignale über einen Schwingkreis leitet, der mit der Lautstärke-Antenne verstimmt wird. Aber das hätte leider fatale Nebenwirkungen. Man würde damit nämlich auch den Pitch-Oszillator verstimmen. Und dann käme es zu einer gegenseitigen Beeinflussung zwischen Lautstärke und Tonhöhe.

Ein Blick in alte Theremin-Schaltungen zeigt einen ganz anderen Weg. Dort gab es einen dritten Oszillator, der ähnlich wie ein Dip-Meter verwendet wird. Das Grid-Dip-Meter (**Bild 3**) wird verwendet, um Resonanzfrequenzen zu messen. Ein Resonanzkreis in der Nähe der Oszillatorspule entzieht dem Oszillatorkreis Energie, so dass die Schwingungsamplitude geringer wird. Das sieht man dann an dem veränderten Gitterstrom. An der Resonanzstelle entsteht ein Dip, d.h. ein besonders kleiner Gitterstrom.

Im Theremin wird die negative Gitterspannung des Oszillators verwendet, um die Verstärkung einer Regelröhre im NF-Zweig zu steuern. Damit umgeht man zugleich die Stabilitätsprobleme, denn die Dip-Frequenz und die NF liegen weit entfernt von der Frequenz der Überlagerungsoszillatoren. Bei einer Röhrenschaltung könnte der frei schwingende Oszillator eine Amplitude von mehr als 10 V erreichen. An einem hochohmigen Gitter-Ableitwiderstand von ca. 1 MΩ könnte man eine Gitterspannung von etwa -10 V erwarten. Wird ein zweiter Schwingkreis lose angekoppelt und genau in Resonanz gebracht, dürfte die Amplitude und damit die negative Gitterspannung vielleicht bis auf 1 V absinken. Das wäre der Grundzustand für volle Lautstärke. Wenn dann jemand den passiven Resonanzkreis verstimmt oder dämpft, wird die negative Gitterspannung wieder stark ansteigen. Mit -10 V lässt sich eine NF-Röhre völlig sperren.

Der erste Versuch mit einem FET-Dip-Meter wurde mit relativ großen Induktivitäten durchgeführt (**Bild 4**). Tatsächlich erreicht die Schaltung eine große Oszillatoramplitude und eine Ausgangsspannung von -4 V. Der Resonanzdip ist vorhanden, allerdings zu schwach. Insgesamt ändert sich die Spannung nur um 0,5 V. Außerdem ist ein Mitzieh-Effekt zu beobachten, der auch beim normalen Dip-Meter auftritt. Die Amplitude ändert sich dann plötzlich an einer der Flanken des Dips.

Tesla-Kopplung

Ein Blick in alte Theremin-Schaltpläne zeigt eine Art Verlängerungsspule in der Antennenzuleitung. Bei genauer Betrachtung funktioniert das wie bei einem Tesla-Transformator. Die Verlängerungsspule hat eine Eigenresonanz auf der Arbeits-

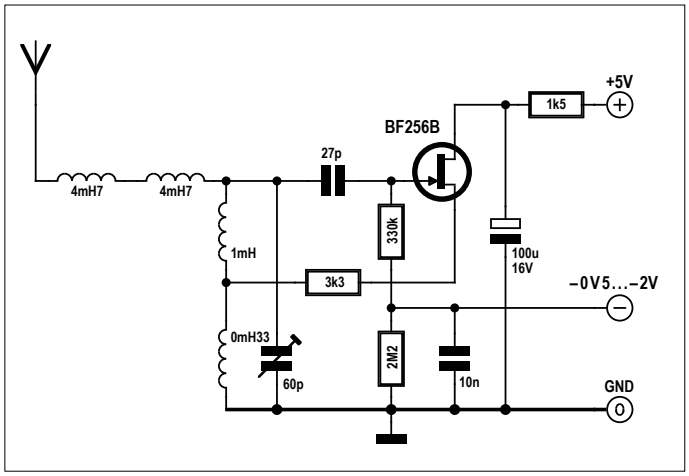


Bild 5. Oszillator mit Antennenspule.

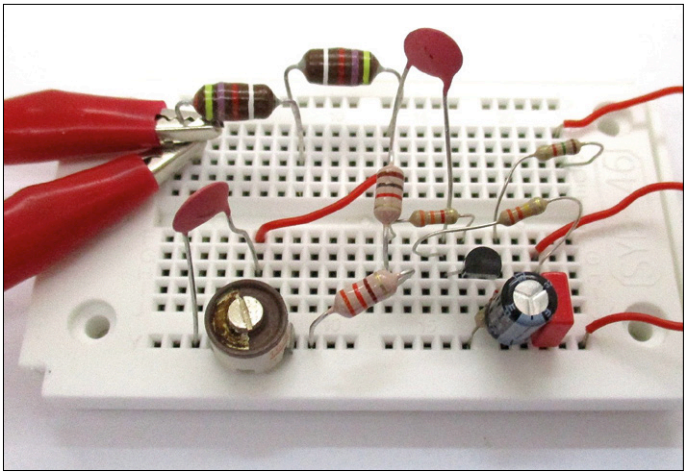


Bild 6. Experimenteller Aufbau des Volume-Oszillators.

frequenz und besitzt ein extrem großes L/C-Verhältnis. Außerdem besteht eine starke Kopplung zum Oszillator. Diesmal wird der Oszillator (**Bild 5**) auf die Tesla-Spule abgestimmt, deren Resonanzfrequenz sich auch bei kleinsten Kapazitätsänderungen stark verstimmt.

Im Resonanzfall hat man an der Antenne eine mehrfach überhöhte HF-Spannung. Der Sekundärkreis saugt dabei viel Energie aus dem Oszillator, so dass dessen Amplitude deutlich abnimmt. Der Gitterspannungs-Dip reicht bis ungefähr -0,5 V. Bei starker Verstimmung oder Dämpfung bei direkter Berührung der Antenne fällt dieser Effekt weg und die Gate-Spannung sinkt um bis zu 2 V. Die Resonanz ist gut einstellbar, es gibt keinen Mitzieheffekt mehr und die Regelspannung ist ausreichend groß. Zum experimentellen Aufbau eines solchen Oszillators reicht eine Steckplatine (**Bild 6**). Als Antenne eignet sich eine Kabelschleife aus einem Krokodilkabel.

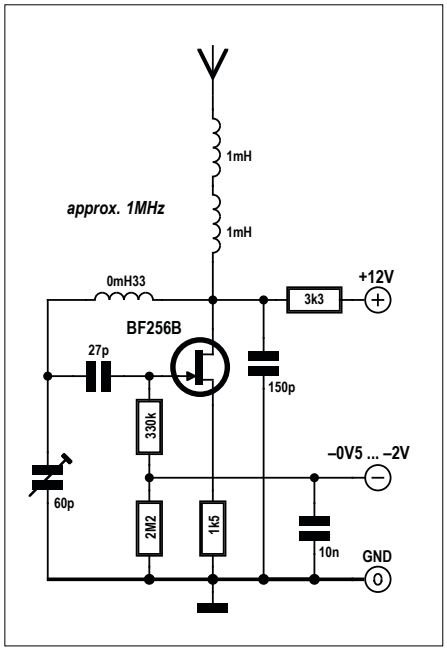


Bild 7. Alternative Oszillatorschaltung.

Vereinfachte Oszillator-Schaltung

Mit der veränderten Oszillator-Schaltung nach **Bild 7** wird eine angezapfte Schwingkreis-Induktivität unnötig. Stattdessen arbeitet der Schwingkreis mit einem kapazitiven Spannungsteiler. Man kann über das Verhältnis der Teilkapazitäten die Impedanz auf der Gate- und Drain-Seite anpassen. Diesmal arbeitet der Oszillator bei rund 1 MHz. Hier liegt auch die Eigenresonanz der beiden Verlängerungsspulen. Wie bei den vorigen Versuchen steigt die Oszillator-Amplitude an, wenn der obere Kreis verstimmt oder gedämpft wird. Die Gate-Spannung sinkt dann bis auf -2 V.

Das vollständige Theremin

Die vollständige Schaltung (**Bild 8**) enthält auch den zweiten Oszillator, den Mischer und eine Schaltung für die Lautstärke-Antenne (Volume). Der zweite Oszillator wird mit einem Keramikoszillator auf 470 kHz stabilisiert. Damit muss auch der abstimmbare Oszillator auf diese Frequenz eingestellt werden können. Zur Abstimmung wird ein Drehko verwendet.

Beide frei schwingenden Oszillatoren verwenden jetzt Spulen mit 1 mH. Bei 470 kHz braucht man dazu 115 pF. Die Kapazität der Antenne wurde mit 5 pF gemessen. Bei Annäherung der Hand auf wenige Zentimeter stieg die Kapazität auf 6 pF. Diese kleine Änderung von nur einem Pikofarad führt rein rechnerisch zu einer Frequenzänderung von 2 kHz. Da hätte man nur einen Spielbereich sehr nahe an der Antenne. Die Verlängerungsspule mit 4,7 mH bewirkt, dass die Kapazitätsänderung am unteren Spulenende stark vergrößert wird.

Auch der Dip-Oszillator wird nun mit 1 mH betrieben. Für die Verlängerungsspule mit der Volume-Antenne wurde eine Eigenresonanz bei 600 kHz gefunden. Der frei schwingende Oszillator muss daher auf etwa 600 kHz abgestimmt werden, was eine Kreiskapazität von insgesamt 70 pF erfordert. Bei 600 kHz entzieht der Antennenkreis dem Oszillator so viel Energie, dass die Regelspannung auf -2,5 V einbricht. Verstimmt man den Antennenkreis durch Annäherung mit der Hand, steigt die Oszillator-Amplitude so weit an, dass die Regelspannung -3,5 V erreicht.

Die Antennenspule mit 4,7 mH kommt mit insgesamt 26 pF in Resonanz bei 600 kHz. Der Großteil dieser Kapazität geht auf die Eigenkapazität der Wicklung zurück, nur der kleinere Teil auf die Antenne. Eine Änderung von einem Pikofarad bei Annäherung der Hand verstimmt die Resonanzfrequenz um 12 kHz. Das reicht für den gewünschten Dip aus.

Als letztes wurde der Gain-Amp entwickelt. Die Verstärkung wird über die Gate-Spannung geregelt. Bei -3,5 V ist der FET vollständig gesperrt, das Ausgangssignal ist auf Null heruntergesteuert. Bei -2,5 V am Gate hat man bereits eine hohe Verstärkung. Der steuerbare Verstärker wird in Gate-Basisschaltung betrieben. Das Signal wird am Source-Anschluss zugeführt und am Drain aus-

gekoppelt. Der Dip-Oszillator muss so eingestellt werden, dass er die passende Regelspannung liefert, um bei Annäherung der Hand das Signal völlig herunterregeln zu können. Zuletzt wurden noch die Tiefpass-Filter im NF-Bereich angepasst, so dass möglichst geringe HF-Reste am Ausgang stehen.

Der Aufbau auf der durchgehenden Kupferfläche einer unbeschichteten Platine (**Bild 9**) hat den Vorteil einer guten Masseverbindung und Entkopplung der einzelnen Stufen. Wenn irgendwo Masse gebraucht wird, ist sie auf dem kürzesten Weg erreichbar. Diese Aufbautechnik mit kleinen Platinen-Stückchen (**Bild 10**) ist auch für komplexere HF-Projekte auf höheren Frequenzen brauchbar. Eine weitere wichtige Erfahrung aus diesem Projekt ist, dass ein Oszillator mit einem JFET sehr frequenzstabil und unempfindlich gegen Schwankungen der Betriebsspannung ist.

Bei allen Vorversuchen war die Schaltung geerdet, weil sie über ein geerdetes Labornetzteil versorgt wurde. Als aber alles fertig in das Gehäuse eingebaut (**Bild 11**) und die Versorgung auf Batterien umgestellt wurde, gab es zunächst Probleme mit einer starken Beeinflussung zwischen Pitch und Volume. Wenn das Gerät geerdet wird, sind diese Probleme sofort verschwunden. Ist ja auch klar, ohne Erdung schaltet man zwei Handkapazitäten in Reihe, da müssen sie sich ja beeinflussen. Aber mit einem Erdanschluss ist das Instrument sehr brauchbar. Und jetzt kann musiziert werden. Man gewöhnt sich relativ schnell daran, rechts die Tonhöhe, links die Lautstärke. ⏮

(160153)

Weblinks

[1] www.elektormagazine.de/160165

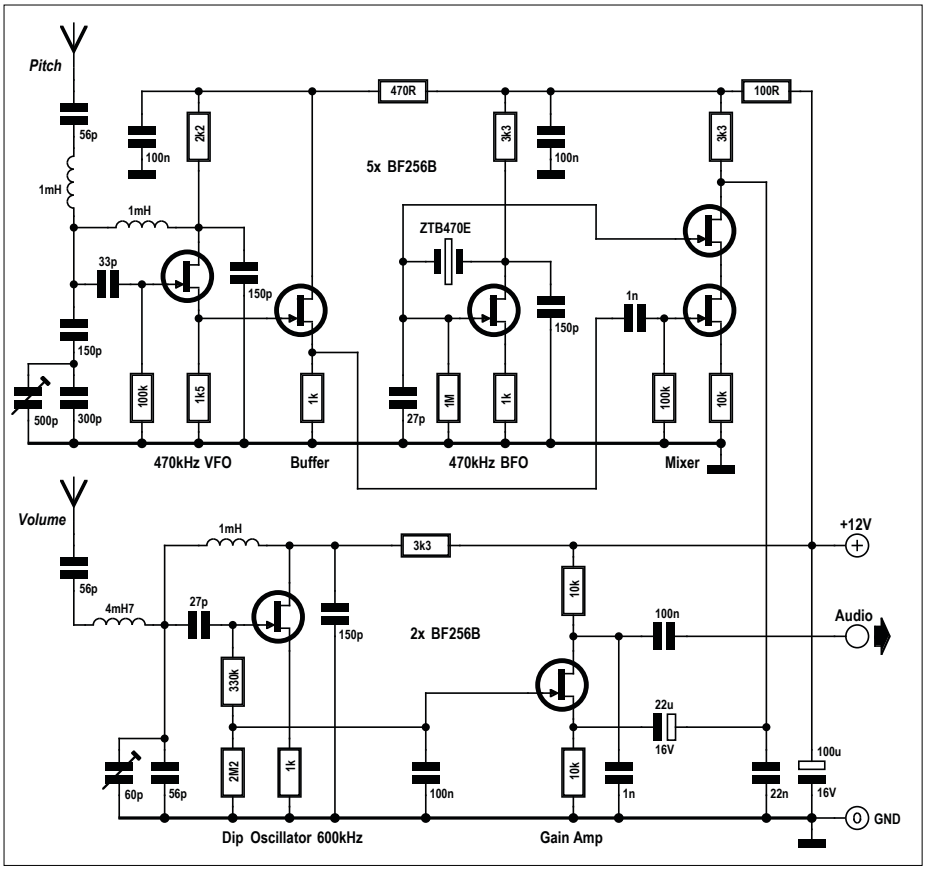


Bild 8. Das vollständige JFET-Theremin.

Die Schaltung kann auf einer unbeschichteten Platine aufgebaut werden.

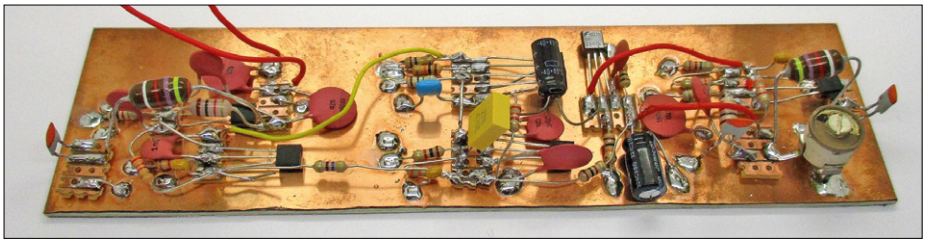


Bild 9. Alle Stufen auf einer Leiterplatte.

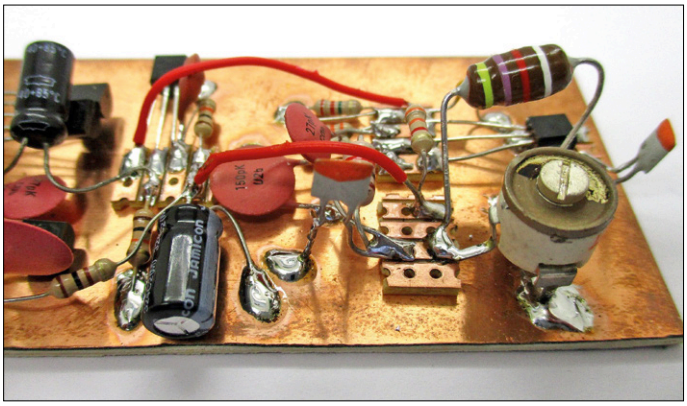


Bild 10. Der Volume-Oszillator im Detail.

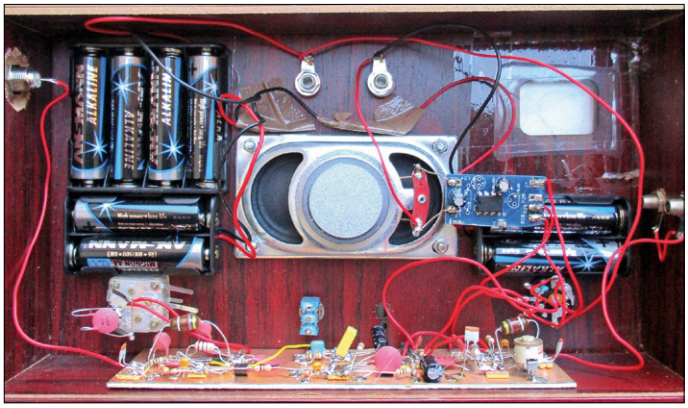
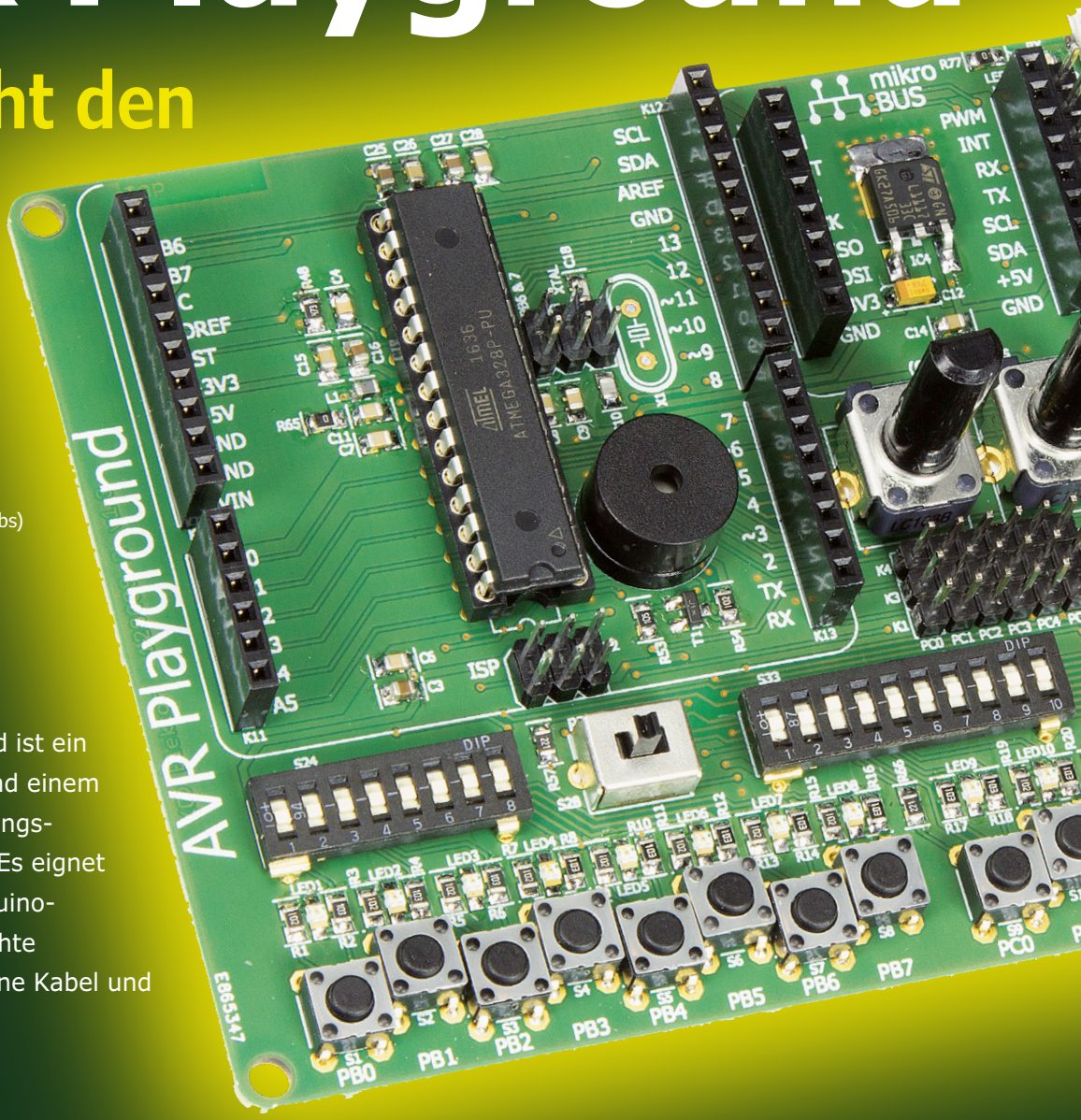


Bild 11. Einbau in ein Gehäuse.

AVR Playground

Vereinfacht den Umgang mit Arduino

By **Clemens Valens** (Elektor Labs)



Das hier vorgestellte Board ist ein Hybrid aus Arduino Uno und einem konventionellem Entwicklungs-Board für Mikrocontroller. Es eignet sich besonders für die Arduino-Entwicklung – ohne schlechte Verbindungen, abgegangene Kabel und ähnliche Nervtötereien.

Bevor Arduino so bekannt wurde, hatten Entwicklungboards für Mikrocontroller etliche Peripherie wie Taster, LEDs, Display, Potis (für analoge Werte), Erweiterungsstecker und anderes mehr auf ihrer Platine. Dazu gehörte natürlich auch ein ordentliches Netzteil. Diese Boards zielten auf eine einfache Art des Erlernens der Mikrocontroller-Programmierung ab, ohne dass man immer wieder Bauteile bestücken oder entlöten musste. Hinter AVR Playground – eine Anspielung auf das unter Arduino Playground bekannte Arduino-Forum – stecken genau die gleichen Überlegungen, weshalb dieses Board viele Funktionen mitbringt, die Arduino-Anwender schätzen.

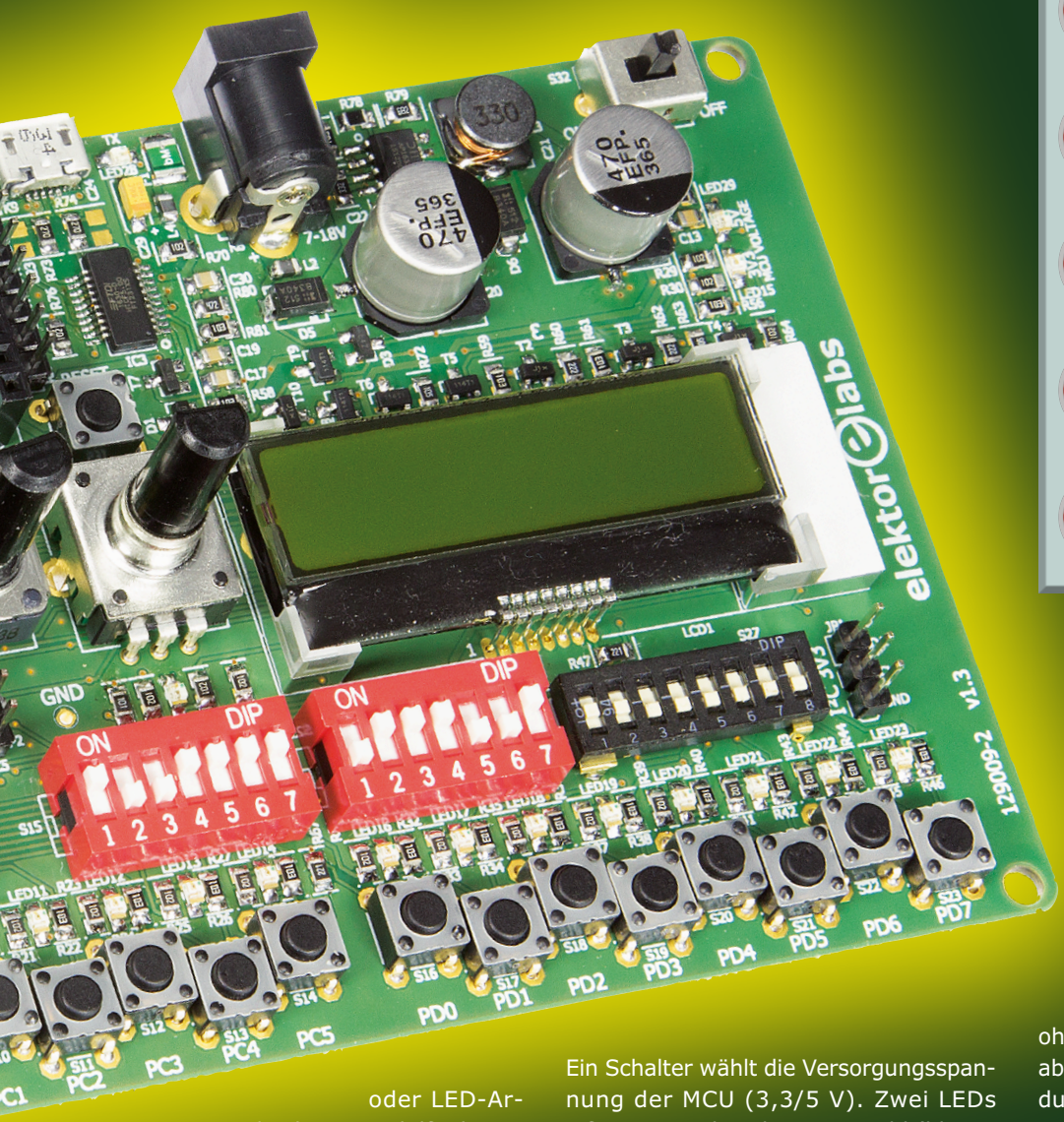
Das Board ist grob in vier Bereiche aufgeteilt (**Bild 1**). Von oben nach unten:

MCU, Erweiterungsstecker, USB/Seriell-Konverter und Netzteil;
Benutzer-Schnittstellen wie Buzzer, Drehknöpfe und ein Display;
DIP-Switches zur Konfiguration;
Taster und LEDs.

Wer an der Schaltung, der Platine oder der Stückliste von AVR Playground interessiert ist, der sei auf [1] verwiesen.

Erweiterungsstecker

Auf der Platine oben links finden sich der Mikrocontroller und die zum Arduino Uno kompatiblen Buchsenleisten. Dieser Teil entspricht exakt einem Arduino Uno. Rechts davon ist ein Bereich mit einem mikroBus-Expansion-Slot (ein Standard, der von der serbischen Firma MikroElektronika entwickelt wurde). Dank ihm erhält man Zugang zu hunderten preiswerter Erweiterungs-Boards, vom GPS-Empfänger über allerlei Sensoren



INFOS ZUM PROJEKT:

- Mikrocontroller
- Arduino
- ATmega328
- Programmierung

Einsteiger
➡ Fortgeschrittener
Experte

Etwa vier Stunden

Löten von SMDs

Rund 75 €

oder LED-Arrays bis hin zu Mobilfunk-Modems. Die Kombination von mikroBus und den Headern für Arduino-Uno-kompatible Shields erlaubt die direkte Verwendung vielfältigster Zusatz-Hardware.

Taster, LEDs & DIP-Switches

Der untere Teil der Platine ist übersät mit Tastern und LEDs, die mit allen verwendbaren Ports des Mikrocontrollers verbunden sind. Die Taster generieren logische Pegel und die LEDs geben eine optische Rückmeldung. Die DIP-Switches darüber dienen zur Konfiguration:
des Abschlusses der GPIO-Leitungen (Pull-up, Pull-down oder offen);
des logischen Pegels eines Tastendrucks (high oder low);
der Verwendung von LEDs und sonstiger Peripherie (angeschlossen oder abgetrennt).

Ein Schalter wählt die Versorgungsspannung der MCU (3,3/5 V). Zwei LEDs informieren über diese Auswahl (blau = 3,3 V; rot = 5 V). Die Details finden sich in den **Tabellen 1** bis **5**.

Stromversorgung

Die Stromversorgung ist stabil und robust ausgelegt. Sie ist gegen Kurzschluss und zu hohe Temperaturen geschützt und kann 5 V mit einem Strom von 1 A liefern. Hierzu muss dieser Teil allerdings selbst von einem Steckernetzteil versorgt werden, das minimal 7 V und ausreichend Strom liefert. Man kann das Board natürlich auch via USB versorgen, ist dann aber auf 500 mA beschränkt, wenn man den versorgenden PC nicht schädigen will. Aus den 5 V werden mit Hilfe eines LDO-Reglers belastbare 3,3 V gemacht. Mit einem weiteren Schalter (PWR) kann man das restliche Board stromlos machen, sodass man es in Ruhe und Sicherheit neu konfigurieren kann, ohne es von einem angeschlossenen PC abzutrennen und z.B. die serielle Verbindung zu verlieren.

USB/Seriell-Konverter

Der USB/Seriell-Konverter bietet nicht nur eine 5-V-Quelle zur Versorgung des Boards, sondern dient auch als Programmier-Interface für Arduino-Sketches. Darüber hinaus steht so auch für Anwendungen ein USB-kompatibler serieller Port zur Verfügung. Man kann ihn auch abtrennen und so die verwendeten Port-Leitungen der MCU für andere Zwecke freimachen oder aber den Konverter an andere Pins anschließen.

Benutzer-Schnittstellen

In der Mitte des Boards sitzen für den praktischen Einsatz typische Bauelemente wie ein Buzzer, zwei Potis und ein Drehgeber sowie ein alphanumerisches Display. Zusammen mit den Tastern und den LEDs kann man damit komplett lötfrei praktische Anwendungen realisieren.

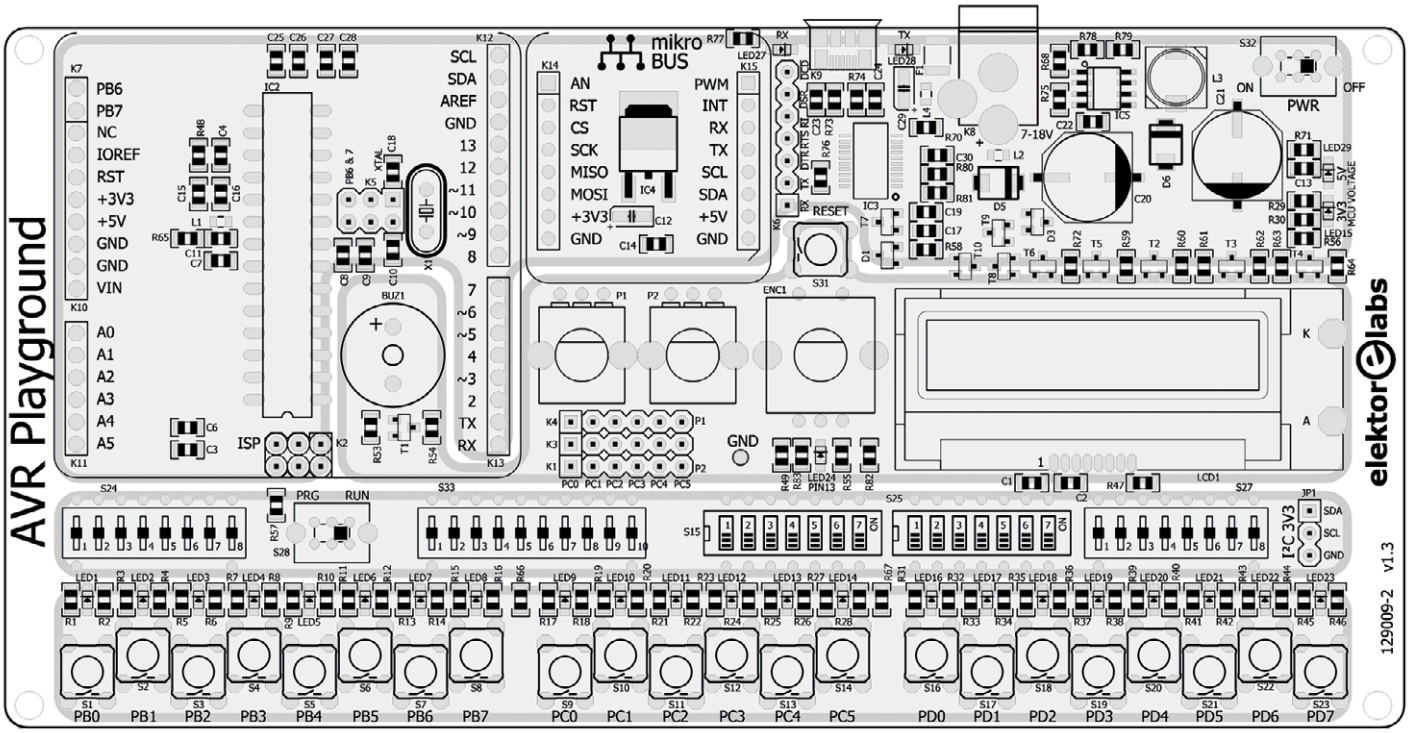


Bild 1. Die Platine von AVR Playground ist in vier Zonen aufgeteilt.

Natürlich kann man mit einem Arduino Uno oder Prototyping-Board sowie einem ordentlichen Drahtverhau im Prinzip das Gleiche erreichen, aber AVR Playground ist sehr viel komfortabler und deutlich zuverlässiger.

Tabelle 1. Konfiguration mit DIP-Switch S15.

S15	Funktion	Ein	Aus
1	Buzzer	Nicht verbunden	Verbunden mit PB1
2	LEDs Port B	Nicht verbunden	Verbunden mit GND
3	LEDs Port C	Nicht verbunden	Verbunden mit GND
4	LEDs Port D	Nicht verbunden	Verbunden mit GND
5	USB/Seriell RXD	Nicht verbunden	Verbunden mit PD1
6	USB/Seriell TXD	Nicht verbunden	Verbunden mit PD0
7	USB/Seriell DTR	Nicht verbunden	Verbunden mit Reset

Tabelle 2. Konfiguration mit DIP-Switch S25.

S25	Funktion	Ein	O Aus n
1	MCU-Versorgung	5 V	3,3 V
2	„Arduino LED“	Nicht verbunden	Verbunden mit PB5
3	LCD SDA	Nicht verbunden	Verbunden mit PC4
4	LCD SCL	Nicht verbunden	Verbunden mit PC5
5	Nicht verwendet		
6	LCD-Beleuchtung	Ein (wenn S25-7 auf „ein“)	Verbunden mit PD7
7	LCD-Beleuchtung	Aus	Steuerbar

Tabelle 3. Konfiguration mit DIP-Switch S24.

S24	Port	Unten	Mitte	Oben
1	PB0	Pull-down	Offen	Pull-up
2	PB1	Pull-down	Offen	Pull-up
3	PB2	Pull-down	Offen	Pull-up
4	PB3	Pull-down	Offen	Pull-up
5	PB4	Pull-down	Offen	Pull-up
6	PB5	Pull-down	Offen	Pull-up
7	PB6	Pull-down	Offen	Pull-up
8	PB7	Pull-down	Offen	Pull-up

LCD

Das Display hat eine Hintergrundbeleuchtung und ist via I²C (statt des üblichen parallelen 8-Bit- oder des semi-parallelen 4-Bit-Interfaces) angebunden, weshalb hier nur zwei MCU-Leitungen erforderlich sind. Der Nachteil ist, dass man einen Treiber benötigt. Doch hierfür wurde im Board-Paket schon Sorge getragen (dazu später mehr).

Drehknöpfe

Mit zwei Potis kann man der MCU analoge Signale zuführen. Über Jumper können sie an jeden der sechs analogen MCU-Eingänge gelegt werden, ohne zu riskieren, die Potis an den gleichen Eingang zu legen.
Der Drehgeber ist technisch fast das Gleiche wie (bis zu) drei Taster, weshalb er parallel an PD3, PD4 und PD5 angeschlossen ist.

Arduino-LED

Diese LED ist an PB5 (Pin 13) des Arduino Uno angeschlossen und wird sehr häufig in Sketches eingesetzt, weshalb sie auch bei AVR Playground vorhanden ist. Sie sitzt unterhalb des Drehgebers.

Taktfrequenz & Reset

Der Standard-Taktgeber für AVR Playground ist der MCU-interne 8-MHz-RC-Oszillator. Damit läuft die MCU unabhängig

von der Versorgung mit 5 V oder 3,3 V immer innerhalb ihrer Spezifikation. Da standardmäßig kein Quarz bestückt ist, stehen die beiden MCU-Leitungen PB6 und PB7 für eigene Applikationen zur Verfügung. Bei Bedarf kann natürlich auch ein Quarz bestückt und über zwei Jumper mit der MCU-verbunden werden.
Normalerweise fungiert die Leitung PC6 als Reset Eingang der MCU. Diese Funktion kann man durch Setzen der RSTDSBL-Fuse der CPU deaktivieren. Dieses Vorgehen ist aber nicht empfehlenswert, da dann die MCU-Programmierung via serielle Schnittstelle **und** ISP-Connector nicht mehr tut. Dann ist der einzig verbliebene Weg zur Programmierung die Entfernung der MCU vom Board und die Verwendung eines sogenannten Parallel-Programmers. Da dies reichlich unkomfortabel ist, wird PC6 nicht wie andere Leitungen behandelt. Es ist also keine LED angeschlossen, aber natürlich ein Taster für eine Spezialfunktion: Reset.

Tabelle 4. Konfiguration mit DIP-Switch S33.

S33	Port	Unten	Unten	Oben
1	Pegel Taster Port B	Low	Nicht verbunden	High
2	Pegel Taster Port C	Low	Nicht verbunden	High
3	Pegel Taster Port D	Low	Nicht verbunden	High
4	PC0	Pull-down	Offen	Pull-up
5	PC1	Pull-down	Offen	Pull-up
6	PC2	Pull-down	Offen	Pull-up
7	PC3	Pull-down	Offen	Pull-up
8	PC4	Pull-down	Offen	Pull-up
9	PC5	Pull-down	Offen	Pull-up
10	Nicht verwendet			

Tabelle 5. Konfiguration mit DIP-Switch S27.

S27	Port	Unten	Mitte	Oben
1	PD0	Pull-down	Offen	Pull-up
2	PD1	Pull-down	Offen	Pull-up
3	PD2	Pull-down	Offen	Pull-up
4	PD3	Pull-down	Offen	Pull-up
5	PD4	Pull-down	Offen	Pull-up
6	PD5	Pull-down	Offen	Pull-up
7	PD6	Pull-down	Offen	Pull-up
8	PD7	Pull-down	Offen	Pull-up

AVR Playground installieren

Auch wenn man AVR Playground natürlich ohne jede vorinstallierte Software verwenden kann, ist das nicht sinnvoll. Stattdessen gibt es sogar mehr Software als üblich, denn da einige Funktionen des Boards nicht von der Arduino IDE unterstützt werden, haben wir Libraries erstellt, mit denen sich die Verwendung der Peripherie dieses Boards vereinfacht. Es gibt gelegentlich Gründe, nicht noch ein weiteres Board zur Arduino IDE hinzuzufügen, weshalb man den Board-Typ „Arduino Pro or Pro Mini“ mit dem Prozessor (via Menü „Tools → Processor“) „ATmega328 (3.3 V, 8 MHz)“ verwenden kann. Die Spannung ist unwichtig - relevant ist nur die ausgewählte Taktfrequenz.
Doch auch das Hinzufügen eines neuen Boards zur IDE ist nicht sehr schwierig. Es gibt sogar ein spezielles Tool dafür: den „Boards Manager“ (via Menü „Tools → Boards“).
Dieser Manager erlaubt die Installation, das Update und das Entfernen von Boards von Fremdherstellern. Damit das klappt, muss der Fremdhersteller ein sogenanntes „Boards Package“ zur Verfügung stellen, das der IDE sagt, was alles für dieses Board heruntergeladen und verwendet werden soll.
Die Installationsprozedur für das Boards Package von AVR Playground ist zwar simpel, erfordert jedoch eine Internet-Verbindung. Los geht es mit dem Menü „File“ und dem Dialog „Preferences“ der Arduino IDE (siehe **Bild 2**, Arduino Version 1.6.13 oder neuer von arduino.cc; **nicht** 1.7.x von arduino.org). Man nutzt den QR-Code von **Bild 3** oder verwendet die nachfolgende URL unter

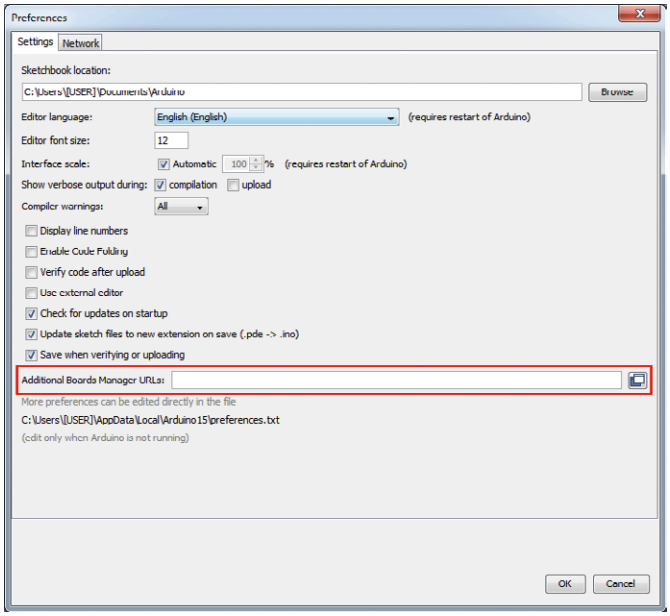


Bild 2. Hier gibt man den URL für das Boards Package zu AVR Playground ein.

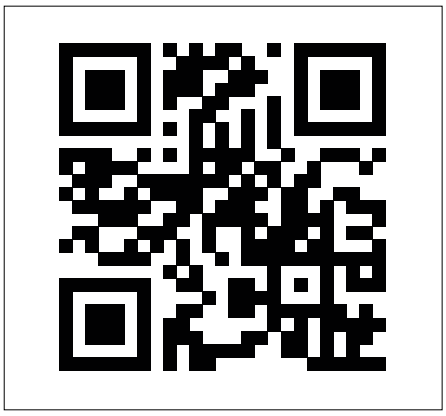


Bild 3. Mit diesem QR-Code kann man Tippfehler vermeiden. Die resultierende URL wird per copy-&-paste in das Feld „Additional Boards Manager URLs“ des Dialogs „Preferences“ übertragen.

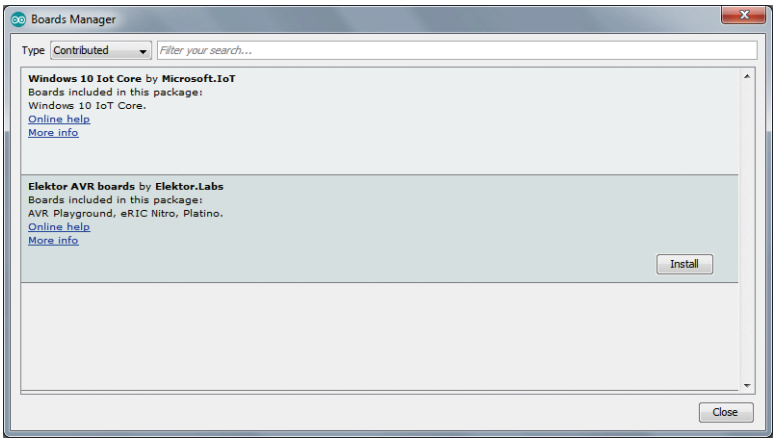


Bild 4. Nachdem die IDE das Boards Package für AVR Playground gefunden hat, kann man es installieren.

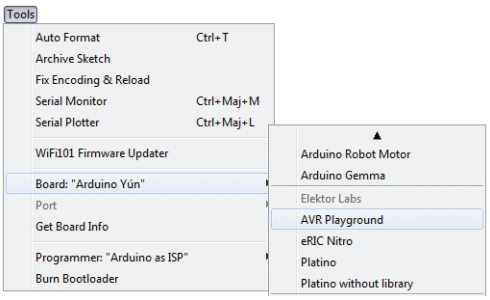


Bild 5. Nach der Installation des Boards Package scrollt man durch die Liste der Boards und wählt AVR Playground aus.

„Additional Boards Manager URLs“ des Dialogs „Preferences“:

```
https://raw.githubusercontent.
com/ElektorLabs/arduino/master/
package_elektor_boards_index.json
```

Anschließend schließt man den Dialog. Jetzt öffnet man den Boards Manager (via „Tools → Boards“). In der oberen linken Ecke des sich öffnenden Fensters wählt man „Contributed“ und sucht „AVR Playground“ in der erscheinenden Liste, klickt drauf und dann auf die Schaltfläche „Install“ (**Bild 4**). Die IDE lädt dann die erforderlichen Dateien herunter und legt sie an den richtigen Stellen ab.
Anschließend schließe man das Fenster. Jetzt taucht AVR Playground irgendwo in der Liste des Menüs „Boards“ unter dem Titel „Elektor Labs“ (**Bild 5**) auf. Selbstverständlich sollte AVR Playground an den PC angeschlossen sein, bevor man seinen „Port“ auswählen kann.

Tabelle 6. Ports mit ihren Pins und jeweiligen Funktionen im Spiel. Zehn GPIO-Pins werden nicht verwendet, weshalb eigentlich auch eine (preiswertere) MCU mit weniger Pins ausreichen würde.

Port B	Funktion	Port C	Funktion	Port D	Funktion
PB0	Nicht verwendet	PC0	Nicht verwendet	PD0	RXD
PB1	Buzzer	PC1	Nicht verwendet	PD1	TXD
PB2	Nicht verwendet	PC2	Nicht verwendet	PD2	LED0
PB3	Nicht verwendet	PC3	Nicht verwendet	PD3	LED1
PB4	Nicht verwendet	PC4	LCD SDA	PD4	LED2
PB5	Nicht verwendet	PC5	LCD SCL	PD5	LED3
PB6	Nicht verwendet			PD6	LED4
PB7	Nicht verwendet			PD7	LED5

Spiele-Programmierung

Wir haben eine Fun-App programmiert, welche die Funktionen des Boards verwendet. Sie ist von einem einfachen Spiel namens „Simon Says“ inspiriert, das in den frühen Tagen der Mikrocontroller-Ära sehr beliebt war. Das originale Spiel gab es früher in einem schwarzen Kunststoffbehälter, der vier große und beleuchtete Taster in den Farben rot, blau, grün und gelb enthielt. Der Computer spielt eine Zufallssequenz, wobei jede Farbe von einem Ton begleitet war. Die Aufgabe des Spielers ist, eine gerade vorgegebene Sequenz zu wiederholen. Bei einem Fehler beginnt das Spiel von vorne. Schafft es der Spieler, wird die nächste Sequenz um ein Element verlängert. Das Spiel ist unter Nerds nach wie vor beliebt. Man kann daher leicht Versionen für PC, Smartphone oder Browser finden.
AVR Playground hat alles „an Board“, was für dieses Spiel erforderlich ist: LEDs, Taster, Buzzer und natürlich auch die Intelligenz in Form eines Mikrocontrollers. Darüber hinaus kann das Spiel durch die Einbeziehung des Displays um Anweisungen, den High Score oder andere Angaben erweitert werden. Es gibt zwar keine farbige Beleuchtung, aber farbige Klebefilmstreifen oder farbiges Papier helfen. Nachfolgend werden relevante Teile des Demo-Programms besprochen. Der komplette Sourcecode steht kostenlos unter [1] zum Download zur Verfügung.

Pin-Belegung

Zunächst muss klar sein, welcher Pin für welche Funktion verwendet werden soll (**Tabelle 6**). Dies beeinflusst auch die DIP-Switches für die Konfiguration (siehe **Tabellen 7** bis **11**). Die gewählten Einstellungen aktivieren die Display-Beleuchtung, und da die zugehörigen Taster auf „high“ eingestellt sind, leuchtet automatisch die zugehörige LED, wenn ein Taster betätigt wird. Damit diese optische Rückmeldung funktioniert, müssen an den entsprechenden Eingängen Pull-down-Widerstände aktiviert sein.

Die LED-Ansteuerung...

...ist etwas komplexer als üblich, da parallel zu den LEDs ja Taster angeschlossen sind. Um flexibler zu sein, wurde eine Lookup-Tabelle angelegt; ändert man die Werte, so kann man auch andere LEDs verwenden **(Listing 1)**.

Tabelle 7. Generelle Konfiguration von S15.		
S15	**Funktion**	**Position**
1	Buzzer	Ein
2	LEDs Port B	Aus
3	LEDs Port C	Aus
4	LEDs Port D	Ein
5	USB/Seriell RXD	Ein
6	USB/Seriell TXD	Ein
7	USB/Seriell DTR	Ein

Tabelle 8. Konfiguration von S25 für das Spiel.		
S25	**Function**	**Position**
1	MCU-Versorgung	Aus
2	„Arduino LED“	Ein
3	LCD I²C SDA	Ein
4	LCD I²C SCL	Ein
5	Nicht verwendet	
6	LCD Backlight	Aus
7	LCD Backlight	Ein

Tabelle 9. S24 konfiguriert Port B, der im Spiel nicht verwendet wird.		
S24	**Port**	**Position**
1	PB0	Mitte
2	PB1	Mitte
3	PB2	Mitte
4	PB3	Mitte
5	PB4	Mitte
6	PB5	Mitte
7	PB6	Mitte
8	PB7	Mitte

Tabelle 10. S33 konfiguriert Port C und den Pegel bei betätigten Tastern.		
S33	**Funktion**	**Position**
1	Pegel Taster Port B	Mitte
2	Pegel Taster Port C	Mitte
3	Pegel Taster Port D	Oben
4	PC0	Mitte
5	PC1	Mitte
6	PC2	Mitte
7	PC3	Mitte
8	PC4	Mitte
9	PC5	Mitte
10	Nicht verwendet	

Tabelle 11. S27 konfiguriert Port D.		
S27	**Port**	**Position**
1	PD0	Mitte
2	PD1	Mitte
3	PD2	Unten
4	PD3	Unten
5	PD4	Unten
6	PD5	Unten
7	PD6	Unten
8	PD7	Unten

Die Taster-Abfrage...

... basiert auf dem gleichen Prinzip wie bei den LEDs, außer dass die Leitungen als Ein- und nicht als Ausgänge dienen. Aufgrund der Konfiguration der DIP-Switches produziert ein betätigter Taster ein logisches „high“.

Ein typisches Problem von Tastern liegt in ihrer mechanischen Natur: Sie brauchen eine kurze Zeit, bis der Kontakt stabil schließt. Aus diesem Grund ist eine Entprellung erforderlich. Wenn es wie im angeführten Spiel auf genaues Timing nicht ankommt, dann gibt es eine einfache Lösung: Man fragt den Eingang einfach nach einer kurzen Wartezeit nochmals ab. Erst wenn er auch beim zweiten Mal „high“ liefert, gilt der Taster als betätigt. Wenn die Abtastrate hoch genug ist (z.B. 10 Hz), wird man kaum einen Tastendruck verpassen. Ein logisches AND („&“) von zwei Abfragen entfernt ungültige Zustände **(Listing 2)**.

Zufallszahlen

Das Spiel benötigt eine Sequenz aus Zufallszahlen (für die Auswahl der leuchtenden LED), die lang genug ist, damit praktisch jeder Spieler irgendwann scheitert. Man kann diese Zahlen einfach in einer Tabelle ablegen. Die Größe dieser Tabelle ist wichtig: Ist sie zu klein, können sich Spieler die Reihenfolge merken und gewinnen zu einfach. Ist sie zu lang, verlieren die Spieler immer und auch das Interesse. Die Tabelle wird am Anfang

Listing 1.

```
#define LED_BUTTON_TABLE_SIZE  (4)
uint8_t led_button_table[LED_BUTTON_TABLE_SIZE] = { 4, 5, 6, 7 };

void led_set(uint8_t nr, uint8_t value)
{
  pinMode(led_button_table[nr],OUTPUT);
  digitalWrite(led_button_table[nr],value);
}
```

Listing 2.

```
uint8_t button_read_all_debounced(void)
{
  uint8_t buttons = button_read_all();
  delay(10); // Wait for any bouncing to stop.
  buttons &= button_read_all(); // AND the second scan.
  return buttons;
}
```

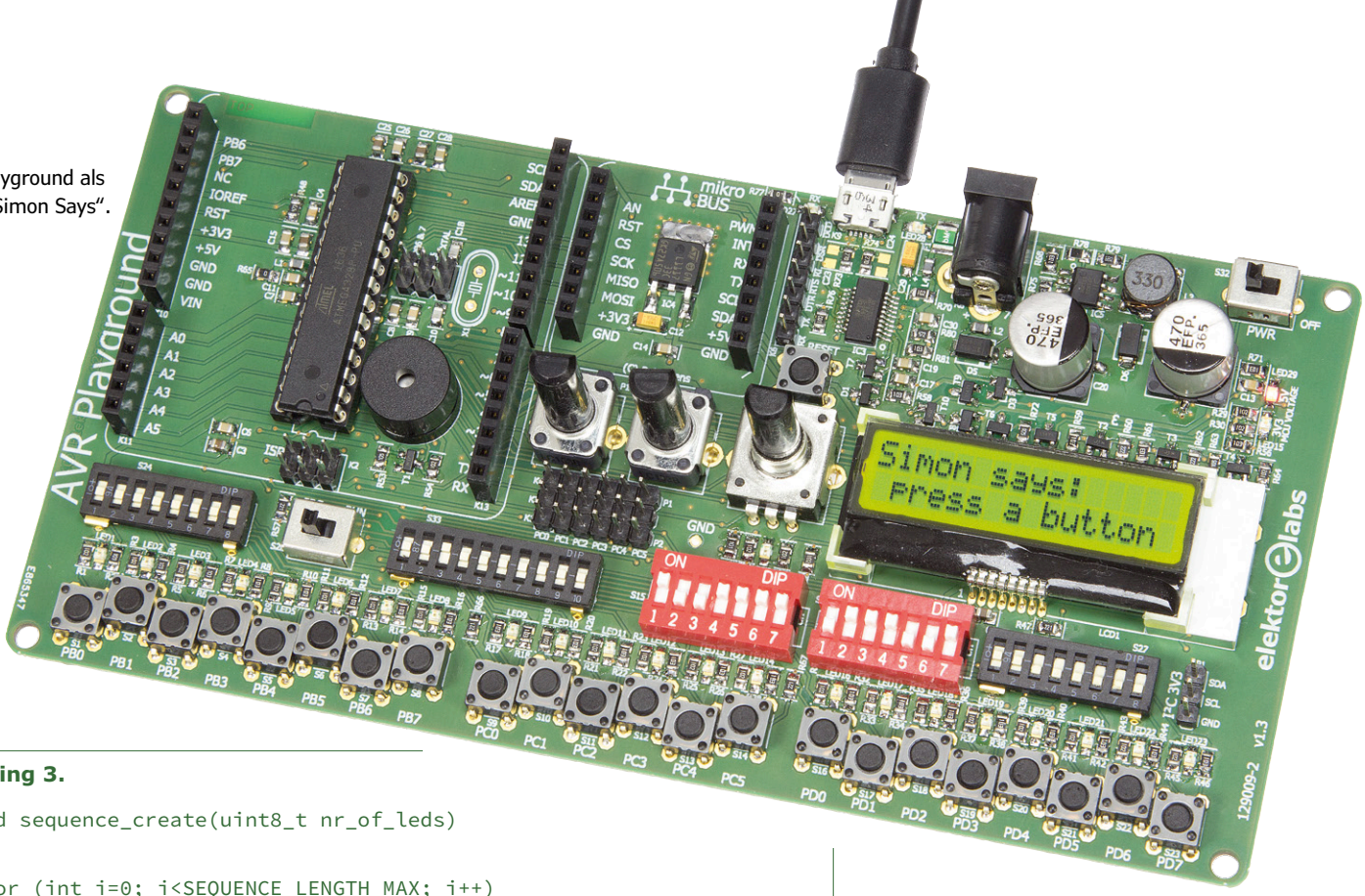


Bild 6.
AVR Playground als Spiel „Simon Says“.

Listing 3.

```
void sequence_create(uint8_t nr_of_leds)
{
  for (int i=0; i<SEQUENCE_LENGTH_MAX; i++)
  {
    sequence[i] = rand()/(RAND_MAX/nr_of_leds + 1);
  }
}
```

Listing 4.

```
void setup(void)
{
  while (button_pressed()==false);
  srand(millis());
  sequence_create(LED_BUTTON_TABLE_SIZE);
  game_round = 1;
}
```

Listing 5.

```
const uint8_t buzzer = 9;
const uint16_t button_sound[LED_BUTTON_TABLE_SIZE] =
{ 554, 659, 880, 1319 }; // C#5, E5, A5, E6 in Hz

#define SEQUENCE_SPEED  (400) /* ms */

void sequence_play(uint8_t len)
{
  for (int i=0; i<len; i++)
  {
    tone(buzzer,button_sound[sequence[i]],SEQUENCE_SPEED/2);
    led_set(sequence[i],true);
    delay(SEQUENCE_SPEED);
    led_set(sequence[i],false);
    delay(SEQUENCE_SPEED/4);
  }
}
```

eines Spiels gefüllt.
Die Funktion `rand` ist für die Generierung von Zufallszahlen im Bereich 0 bis `RAND_MAX` (beim Arduino = 32.767) zuständig. Ein simples Verfahren zur Einschränkung dieser Zahlen auf den benötigten Bereich ist eine Modulo-Division („%“). Mathematisch gesehen ist das aber fragwürdig, da die Zahlen dann unter Umständen nicht mehr gleich verteilt sind. Besser geht das so wie in **Listing 3**.

Die Erzeugung von Zufallszahlen per Code ist aus Prinzip nichtideal, da Code immer deterministisch ist. Damit die Funktion `rand` sauber arbeitet, muss man sie mit einer Zufallszahl initialisieren – schon hat man ein Henne/Ei-Problem an der Backe. Ein häufig genutzter Trick ist die Verwendung der Zeit als Startwert, da die Zeit immer weiterläuft und so fast eine ideale Folge von Zufallszahlen liefert. Leider nicht bei einem Mikrocontroller, der die Zeit bei jedem Start wieder bei „0“ anfangen lässt. Von daher wird hier ein Generator benutzt, der als Startwert die Zeit verwendet, bis zum ersten Mal eine Taste betätigt wird. Wird die Zeit in Millisekunden aufgelöst, wird es kaum je zu reproduzierbar gleichen Startwerten kommen **(Listing 4)**.

Sequenz mit Sound

Bei jeder Spielerunde wird die Sequenz etwas länger, sie muss aber trotzdem jedes Mal von vorne eingegeben wer-

den. Wird die Sequenz zu schnell dargeboten, können Menschen nicht recht folgen und sich somit nicht gut erinnern. Ist das Tempo zu niedrig, wird es langweilig. Für mehr Spielvergnügen wird zu jeder LED via Buzzer (über Pin 9, PB1) ein Ton abgespielt **(Listing 5)**.

Die Tabelle `button_sound` enthält die Frequenzen der Töne in Hz. Die Konstante `SEQUENCE_SPEED` (in ms) bestimmt die Wiedergaberate der Sequenz. Die Funktion `sequence_play` erhält als Argument die Länge der abzuspielenden Sequenz.

Die Bedienung

Wenn eine Sequenz abgespielt wurde, soll der Spieler versuchen, sie zu wiederholen. Sobald der Spieler einen Fehler macht, beendet das Programm die weitere Abfrage von Tasten. Außerdem gibt es noch eine Feinheit zu beachten: Man sollte vermeiden, dass ein längerer Tastendruck als zwei oder mehrere kürzere interpretiert wird **(Listing 6)**.

Spielen

Was jetzt noch fehlt ist die Funktion `loop`, um die Funktionen `sequence_play` und `sequence_read_buttons` zu koppeln, sowie dabei auf die richtige Sequenz-Länge und den Status des Spiels zu achten **(Listing 7)**.

In den Codeabschnitten nach `if` und `else` steht noch nicht allzu viel, das wollen wir Ihnen überlassen. Zu denken wäre etwa an unterschiedliche Töne je nach Spiel-Status, etwa ein netter Klang für „Prima!“ und „Gewonnen“ sowie entsprechend nicht so nett für „Verloren!“. Das Ganze kann man natürlich mit entsprechendem Feuerwerk durch LEDs würzen. Denken Sie sich was aus und erzeugen Sie eigene „special effects“! ◀

(160316)

Weblink

[1] www.elektormagazine.de/160316

Listing 6.

```
bool sequence_read_buttons(uint8_t len)
{
  for (int i=0; i<len; i++)
  {
    uint8_t nr, buttons;

    // Wait for a button press.
    do
    {
      buttons = button_read_all_debounced();
    }
    while (buttons==0);

    // Check that we have a valid button press.
    nr = button_as_number(buttons);
    if (nr!=BUTTON_PRESS_INVALID && nr<LED_BUTTON_TABLE_SIZE)
    {
      // Play sound & wait until it finishes.
      tone(buzzer,button_sound[nr], SEQUENCE_SPEED/2);
      delay(SEQUENCE_SPEED/2);
      // Wait until the player releases the button.
      while (button_pressed()==true);
      // Fail on wrong button.
      if (sequence[i]!=nr) return false;
    }
    else return false;
  }
  return true;
}
```

Listing 7.

```
void loop(void)
{
  sequence_play(game_round);
  if (sequence_read_buttons(game_round)==true)
  {
    if (game_round<SEQUENCE_LENGTH_MAX)
    {
      // Next round, play ‘well done’ tune?
      game_round++;
    }
    else
    {
      // Player wins, play ‘you win’ tune?
      game_start();
    }
  }
  else
  {
    // Game over, play ‘game over’ tune?
    game_start();
  }
}
```

ADVERTORIAL

Debugging und Dekodierung digitaler Kommunikation

Mit dem SmartScope

Von **Riemer Grootjans** (LabNation)

Auch wenn das SmartScope von LabNation vor allem ein voll funktionsfähiges Oszilloskop ist, können Sie acht Tastköpfe anschließen, um digitale Signale zu erfassen. Dieser Artikel erklärt, wie das SmartScope auch als Logikanalysator zu verwenden ist, damit Sie Ihre digitalen Schaltungen prüfen und digitale Protokolle dekodieren können.

Die meisten modernen Schaltungen sind digital: Entweder ist es eine rein digitale Schaltung oder es gibt einen digitalen Kern, der den Analogteil steuert und/oder die Ausgangssignale erfasst. Wenn ein digitaler Schaltkreis nicht so arbeitet, wie er sollte, müssen Sie ermitteln, wann digitale Leitungen High und wann Low sind und zu welchem Zeitpunkt sich ein Zustand ändert.
Die grundlegende Aufgabe eines Logik-Analysators ist es, ein Ereignis und seine Wirkung zu ermitteln. Verbinden Sie einen Tastkopf mit dem Signal, das Sie überwachen möchten, und ein zweites mit einer Triggerleitung. Der Logikanalysator zeichnet das Verhalten beider Signale über einen bestimmten Zeitraum auf, so dass Sie es analysieren können.
Meist findet digitale Kommunikation auf mehreren Leitungen statt, deren gegenseitiges Timing berücksichtigt werden muss. Das SmartScope bietet acht digitale Eingänge und verfügt über die Funktionalität, die genaue Zeit zwischen verschiedenen Pulsen zu messen.
Und schließlich sollen Logikanalysatoren auch eine serielle Kommunikation nach bestimmten Protokollen zwischen Chips überwachen können. Dekoder helfen Ihnen, den bloßen Datenstrom als sinnvolle Befehle oder Werte zu interpretieren.

Schlüsselspezifikationen

Wenn Sie das SmartScope als Logikanalysator einsetzen, müssen Sie einige Spezifikationen beachten.

Anzahl der Eingänge: Zwei Eingänge stellen das absolute Minimum dar (um auf einen Trigger zu warten und dessen Wirkung auf ein Signal festzustellen). Aber es ist meist besser, die Bits mehrerer Eingänge auszuwerten, um zum Beispiel auf einen bestimmten Wert triggern zu können, den die Bits darstellen.

Das SmartScope verfügt auf der Rückseite über acht digitale Eingänge, an die Sie entweder die 0,1”-Dupont-Drähte (kurze Kabel mit Steckhülsen, die jedem SmartScope beiliegen) anschließen können oder aber das spezielle Logic-Analyzer-Kabel von LabNation.

Eingangsspannungsbereich und Schutz: Der Logik-Analyzer des SmartScopes kann digitale High-Pegel zwischen 1,8 V und 5,0 V erfassen. Neuere Chips verwenden meist niedrige Spannungen, um den Stromverbrauch und die durch das Laden und Entladen von Leiterbahnen und Pins verursachte EM-Strahlung zu reduzieren.
Jeder Eingang des SmartScopes enthält Schutzdioden in beide Richtungen, so dass im Falle eines unerwarteten Verhaltens oder Spannungs/ESD-Spikes nichts passiert. Sie müssen nur sicherstellen, dass die kontinuierliche Spannung 5,5 V nicht überschreitet.
Abtastrate: Das SmartScope tastet die Signale mit 100 MS/s ab. Es können Impulse von nur 10 ns Länge dargestellt und das Tastverhältnis eines 500-kHz-Signals mit einer Auflösung von 1 % gemessen werden.
Sample-Speicher/RAM: Kleine Sample-Speicher (<10 K) reichen zwar zur Darstellung einfacher Zusammenhänge aus. Wenn Sie jedoch die vollständige Kommunikation zwischen digitalen Chips erfassen möchten, benötigen Sie viel mehr Speicherplatz. Deshalb ist das SmartScope mit einem RAM ausgestattet, das 4 MSamples/Kanal speichern kann. Nach der Erfassung können Sie sich Zeit nehmen, einen beliebigen Teil der Übertragung zu vergrößern, zu dekodieren und jede Flanke zu überprüfen. Das bringt uns zum nächsten Punkt, dem **Protokoll-Dekoder**: Die Überprüfung jeder einzelnen Flanke einer längeren digitalen Kommunikation kann sehr umständlich, zeitraubend und frustrierend sein. Um dies zu vereinfachen, verfügt die SmartScope-Software über Dekoder für die gängigsten Protokolle wie 1-Wire, UART/RS232, I²C, I²S und SPI. Das Dekodersystem des SmartScopes ist sehr flexibel, so dass Dekoder auch verwendet werden können, um zusätzliche Informationen wie die Gesamtzahl der Flanken oder die Intervalle dazwischen anzuzeigen. Sie können sogar Ihre eigenen Dekoder [1] zusammenbauen!
Die Protokolldekodierung wird wie bei fast allen USB-basierten Logikanalysatoren von der Software abgewickelt. Folglich werden die Daten über eine definierte Zeitspanne erfasst und zum PC übertragen, wo die Software die Daten dekodiert. Auch wenn Sie in den dekodierten Daten suchen können, gibt es keine „robuste“ Möglichkeit, auf sie zu triggern. Dafür müssten die dekodierten Daten in der Hardware verfügbar sein und Logikanalysatoren, die dies können, kosten ein Vielfaches eines vollständigen SmartScopes.

Um zu beginnen...

In diesem Abschnitt wird erläutert, wie Sie mit dem Logic-Analyzer des SmartScopes beginnen und wie Sie die besten Ergebnisse erzielen.
Schließen Sie die Hardware richtig an: Genau wie bei einem Oszilloskop benötigt ein Logikanalysator eine Referenz von 0 V zur Messung. Es muss also mindestens eine Verbindung zwischen den Massen des SmartScopes und des zu testenden Geräts geben. Dann verbinden Sie die Eingänge des SmartScopes mit den Signalen der Testschaltung, deren Daten Sie erfassen möchten.

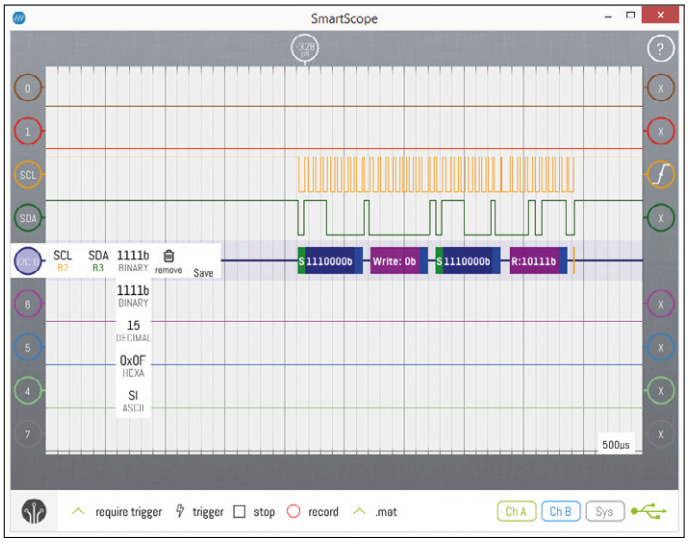


Bild 1. Der Protokoll-Dekoder bei der Arbeit.

Einrichten der Software: Sie schließen das SmartScope an Ihren PC/Tablet/Smartphone an und starten die SmartScope-App. Folgen Sie *Menu → Digital mode*. Standardmäßig erscheinen die acht digitalen Kanäle. Falls Sie nicht alle Kanäle angeschlossen haben und den freien Speicherplatz nutzen möchten, tippen Sie auf den Indikator eines freien Kanals. Wenn Sie „Hide“ wählen, wird der Kanal entfernt und sein Indikator unten rechts auf dem Bildschirm geparkt (**Bild 5**). Tippen Sie einfach nochmals darauf, um diesen Kanal wiederherzustellen.
Konfigurieren der Triggerung: Wenn die Tastköpfe an eine laufende Schaltung angeschlossen sind, dürften wahrscheinlich viele Signale im wilden Durcheinander zu sehen sein. Durch Einrichten einer Triggerbedingung können Sie das SmartScope veranlassen, auf eine bestimmte Kombination der digitalen Signale zu warten, bevor sie erfasst und angezeigt werden. Dazu wird das Symbol innerhalb der Triggeranzeige auf der rechten Seite jedes Signals geändert. In **Bild 1** wartet das SmartScope beispielsweise, dass eine ansteigende Flanke auf D2 erscheint. Die Triggerposition kann nach links oder rechts verschoben werden, so dass Sie sich auf den interessanten Bereich konzentrieren können. Der Bereich kann sehr breit gewählt werden, man kann die Triggerposition außerhalb der Grenzen Ihres Bildschirms oder sogar außerhalb der Grenzen des RAMs einstellen. Die Triggerposition im RAM wird durch die rote vertikale Linie oben im Panorama angezeigt.
Und schließlich gibt es einen Trigger-Modus. Im Modus „Require trigger“ wartet das SmartScope auf eine Triggerbedingung und zeigt nur dann die erfasste Sequenz auf dem Bildschirm an. Das Verhalten im Modus „Normal trigger“ ist identisch, wenn aber innerhalb einer Sekunde keine Trigger-Bedingung festgestellt wurde, erfasst das SmartScope nach Belieben und zeigt die Sequenzen auf dem Bildschirm an. Der „Single trigger“-Modus wartet darauf, dass eine Triggerbedingung eintritt und stellt dann die erfassten Daten in detailliertester Form auf dem Bildschirm dar.

Zugriff auf grundlegende Informationen

Wenn die erfassten Daten auf dem Bildschirm stehen, gibt es viele Informationen, auf die Sie sofort zugreifen können (**Bild 2**).

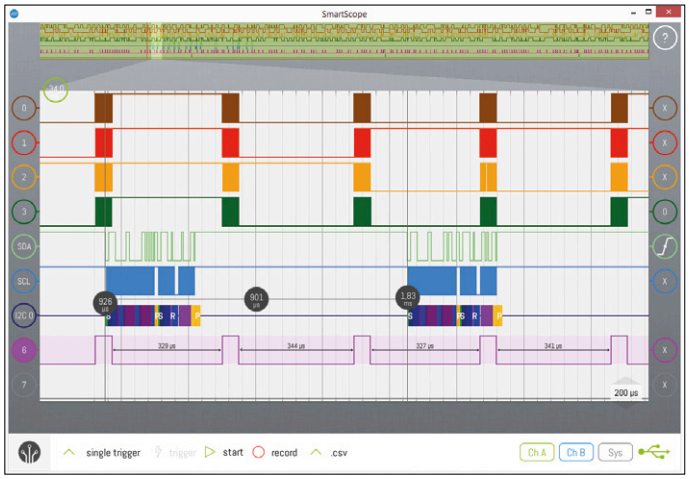


Bild 2. Anschluss eines 5-MP-RGB-Bildsensors. Die unteren Signale sind HSync und VSync, die vier oberen zeigen die momentanen Pixeldaten. D4 und D5 sind die Data- respektive Taktsignale des I²C-Interfaces, wobei ein angefügter Dekoder die Daten direkt dekodiert.

Bild 3. Dekodierter I²C-Befehl.

Intervallzeit: Beim Bewegen der Maus über ein beliebiges Signal wird das Zeitintervall zwischen allen Flanken angezeigt.
Zeitcursor: Vom Rand der Darstellung kann ein Zeitcursor auf das Diagramm gezogen werden. Fügt man einen zweiten Cursor hinzu, erscheint ein Differenz-Cursor (Delta-Cursor), der die Zeitspanne zwischen beiden Cursors anzeigt. Wenn Sie auf diesen Delta-Cursor tippen, wird die Zeitangabe durch die entsprechende Frequenz ersetzt.
Flankendekoder: Um die genaue Zahl der Flanken in einem Intervall zu erfahren, folgen Sie *Menu → Add decoder → Edge counter*. Wählen Sie den gewünschten Kanal und lassen Sie den Dekoder die Flanken zählen. Fügen Sie den Dekoder „Edge intervals" hinzu, so wird die Zeit zwischen jedem Intervall mit einer Auflösung von bis zu 10 ns angezeigt.

Verwendung der digitalen Dekoder

In vielen Systemen tauschen digitale Chips Konfigurationsdaten und/oder Messdaten in nur geringer Geschwindigkeit aus. Ein solcher Kanal ist auch in **Bild 2** vorhanden, wo der Controller eine Nachricht nach dem I²C-Protokoll zu einem RGB-Bildwandler schickt. Dieses Protokoll ist gut definiert. Die I²C-Nachricht beginnt mit einem Byte, das die Adresse des Zielchips enthält und die Mitteilung, ob der Master lesen oder schreiben möchte. Jedes SDA-Bit sollte bei einer steigenden Flanke auf der SCL-Leitung interpretiert werden.
Nun könnten Sie beginnen, die Daten auf dem Bildschirm zu dekodieren, indem Sie den SDA-Wert für jede steigende SCL-Flanke notieren. Besser überlassen Sie der SmartScope-App diese Aufgabe: Folgen Sie einfach *Menu → Add decoder → I2C Decoder*. Der Dekoder wird dem Bildschirm hinzugefügt und die Software versucht zu erkennen, welcher Anschluss SDA und welcher SCL ist. **Bild 3** zeigt eine komplexere Situation mit mehreren Signalen mit niedrigen und hohen Frequenzen, bei der man der Software helfen muss. Wenn das Dekoder-Menü standardmäßig geöffnet ist, drücken Sie den SCL-Knopf und wählen D4, drücken dann den Knopf SDA und wählen D5. Schließlich drücken Sie den Indikator, um das Menü abzuschalten. Jetzt kann der Dekoder die gesamte Kommunikation entschlüsseln.

Ändern des Dekoder-Ausgangstyps

Manchmal möchten Sie die dekodierten Werte als einfache Dezimalzahlen, manchmal als hexadezimale Ziffern darstellen, bisweilen ist auch das binäre oder das ASCII-Äquivalent interessanter. Drücken Sie auf den Signal-Knopf und wählen den gewünschten Ausgangstyp (Bild 1). So können Sie zum Beispiel eine UART-Kommunikation direkt nach ASCII übersetzen.

Bauen Sie Ihren eigenen Dekoder

Das Dekoder-Subsystem der SmartScope-App ist sehr flexibel. Sie können sogar einen eigenen Dekoder in C# schreiben, kompilieren und im gleichen Ordner wie die App [1] ablegen. Dadurch können Sie einen Dekoder für Ihr eigenes Protokoll erstellen oder sogar mehrere Dekoder stapeln, also zum Beispiel einen Dekoder erstellen, der den Ausgang des I²C-Dekoders übernimmt und die Registeradressen in Registernamen übersetzt.

Optimale Ergebnisse

Es kann einige Fummelei an den Messkabeln erforderlich sein, um korrekte Messungen zu erhalten, insbesondere bei hochfrequenten Low/High-Signalen. Da die Dupont-Kabel zwischen SmartScope und dem zu testenden Gerät nicht abgeschirmt sind, kann ein Übersprechen zwischen den Kabeln auftreten, beispielsweise eine steigende Flanke auf einem Kabel einen sehr kurzen Impuls auf einem benachbarten Kabel verursachen. Da sowohl elektrische als auch magnetische Störeinflüsse vorliegen, ist es bei ungeschirmten Kabeln am besten, sie möglichst kurz zu halten und in weitem Abstand voneinander zu verlegen. Das höchstfrequente Kabel verlegt man in der Nähe des Massekabels.
Mit den mitgelieferten Dupont-Kabeln beaufschlagen Sie die zu messenden Signale sogar mit einer kleinen Kapazität. Das Signal kann soweit beeinflusst werden, dass die Kommunikation fehlschlägt!

Das Logic-Analyzer-Kabel von LabNation

In Anbetracht dieser Probleme hat LabNation ein Logic-Analyzer-Kabel (**Bild 4**) für die Verbindung zum AUX-Anschluss auf der Rückseite des SmartScopes entwickelt. Jede der Adern ist als geerdetes Koaxialkabel ausgeführt und so bestens geschirmt. Am Signaleingang ist jedes Kabel in einen kurzen Draht für das Signal selbst und einen weiteren für eine mögliche Erdung aufgeteilt,

Das SmartScope kann auch ein analoges Signal innerhalb der erfassten Daten darstellen.

Bild 4. Das Datenkabel des Logic-Analyzers von LabNation.

was eine optimale Abschirmung ermöglicht. An diesen Stellen befinden sich winzige Platinen mit kleinen Filtern, wodurch der Einfluss des Kabels auf das Signal auf ein Minimum reduziert wird.

Dekodierungsstrategie

Zu den praktischen Anwendungen, bei denen ein Logic-Analyzer ein hervorragendes Hilfsmittel ist, gehört die Erfassung, Dekodierung und Speicherung eines langen Kommunikationsstroms in einer Datei.

Verbinden Sie das zu testende Gerät mit dem Logic-Analyzer, am besten mit dem Logic-Analyzer-Kabel, oder stellen Sie sicher, dass die Signalleitungen entfernt voneinander sind und die Leitung mit der höchsten Frequenz nahe der Masseverbindung liegt. Achten Sie auch auf eine perfekte Masseverbindung!

Stellen Sie die Triggerbedingung ein, typischerweise eine steigende oder fallende Flanke auf einem der Signale. Die Signale, die für den Trigger keine Rolle spielen, werden auf „X“ gesetzt, da sonst die Triggerbedingung niemals erfüllt wird.

Als Nächstes stellen Sie das Panorama (die RAM-Ansicht am oberen Rand des Bildschirms) auf die vollständige Länge der Kommunikation, die Sie analysieren möchten, ein. Normalerweise verschieben Sie dazu die Triggerposition (die vertikale rote Linie) im Panorama nach ganz links. Am besten geht das, wenn möglich, indem man den Analyzer im „Auto triggering“-Modus laufen lässt und die Kommunikation der zu testenden Schaltung ein paar Mal hintereinander belauscht.

Schließlich wählen Sie den Modus „Single trigger“ und warten auf den nächsten Durchlauf der Kommunikation. Sobald der Trigger ausgelöst wird, wird der gesamte Inhalt des RAMs auf den Host übertragen. Dort lassen sich die Signale in aller Ruhe durchsuchen. Sie können auch, wenn Sie es für richtig halten, weitere Dekoder hinzufügen.

Wenn Sie so ein so langes Panorama benötigen, dass die Samplefrequenz unter 100 MS/s sinkt, dann gehen Sie im Menü auf *Menu → System → Acquisition depth → 4MS*. So können Sie die volle Tiefe des RAMs benutzen (es dauert dann auch etwas länger, um den Inhalt zum Host zu übertragen).

Der Mixed Mode ist die Rettung!

In einigen Fällen sehen Sie zwar perfekte Signale auf dem Bildschirm, aber Ihre Chips reagieren trotzdem nicht wie gewünscht. Möglicherweise folgen die Signale einer 3,3-V-Logik, während ein IC 5-V-Signale erwartet und die 3,3-V-Signale gar nicht erkennt. Oder umgekehrt, alles funktioniert prächtig, aber Sie sehen unerwartete Spikes auf dem Logikanalysator-Bildschirm. An diesem Punkt brauchen Sie sowohl einen Logikanalysator als auch ein Oszilloskop! Und genau dies ist ja das SmartScope. Folgen Sie *Menu → Mixed mode*, in dem zum Logikanalysator ein analoger Kanal hinzugefügt wird.

Sowohl der analoge als auch die digitalen Kanäle werden synchron gemessen. Wenn Sie also Ihren analogen Tastkopf an eine der digitalen Messkabel anschließen, können Sie sowohl die analoge Form des Signals als auch das digitale Ergebnis sehen (Bild 5). Der gezeigte Graph beansprucht weniger als 1 % des RAM-Speicherplatzes und liefert dennoch genügend Details zur Darstellung der analogen und digitalen Wellenformen.

(160282)

Weblinks

[1] http://wiki.lab-nation.com/index.php/Creating_your_own_Protocol_Decoder

Bild 5. Mixed Mode: Mit Kanal A wird die Spannung auf der SDA-Leitung analog gemessen.

HomeLab Helikopter

Zusammengestellt von **Clemens Valens** (Elektor-Labor)

Der schönste Platz ist immer an der Theke

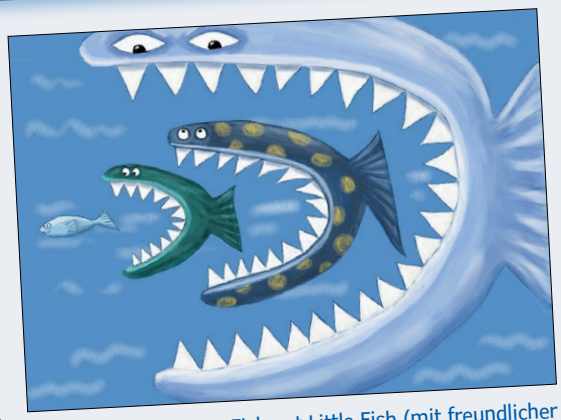
Maker Store Berlin,
Danziger Str. 22, 10435 Berlin

Im Dezember 2016 fand in Berlin eine ziemlich überraschende Veranstaltung statt. Es ging weder um Kunst noch um Sport und schon gar nicht ums Essen, obwohl Getränke und viel Pizza während der Veranstaltung verdrückt wurde – es ging schlicht um die Eröffnung eines Ladens. Doch nicht irgendeiner, sondern der erste Arduino-Store der Welt! Seit ich denken kann, also seit etwa 35 Jahren, gehen Elektronikgeschäfte mit Schaufenstern und Ladentheken pleite, während virtuelle Elektronikgeschäfte überall im Internet auftauchen. Aber die Öffnung eines echten Elektronikladens mit jemandem, der hinter der Theke steht und Sie bedient, ist wirklich ein neues - oder sollte ich revolutionäres sagen? - Konzept. Im neuen Arduino-Maker-Store können Sie – klar - Arduino-Produkte kaufen, aber nicht nur das. Denn da es sich um einen Maker-Laden handelt, sollen dort auch Workshops organisiert werden. Ist das der Beginn eines neuen Trends? Werden schon bald Elektronik-Shops an jeder Straßenecke aus dem Asphalt brechen? Oder wird dieses Geschäft den gleichen Weg gehen wie all seine Vorgänger?

Tipps & Tricks

Q. Wie können Sie feststellen, ob Sie mit Wechsel- oder Gleichstrom arbeiten?
A. *Wenn Sie den Leiter berühren und die Zähne klappern, ist es Wechselstrom. Bei Gleichstrom kleben sie einfach zusammen.*

Halbleiter-Dschungel: fressen und gefressen werden



Big Fish eat Little Fish (mit freundlicher Genehmigung von www.catherineswenson.com)

Wenn Sie die die Elektronik-Industrie beobachten oder auch nur Zeitung lesen, werden Sie die Welle der Fusionen und Übernahmen der letzten Jahre bemerkt haben. Intel schluckte den FPGA-Hersteller Altera, NXP übernahm Freescale, nur um selbst ein paar Monate später von Qualcomm übernommen zu werden. Microchip kaufte den Konkurrenten Atmel, ON Semiconductor verleibte sich Fairchild Semiconductor ein und Analog Devices Linear Technology. Der Bauteil-Distributor Avnet übernahm seinen Kollegen Farnell, der gerade Cadsoft (Eagle) an Autodesk (AutoCAD) verkauft hatte. Mentor Graphics fusionierte mit Siemens und nach dem Brexit wurde ARM von der japanischen Softbank gekauft. Wussten Sie, dass auch Amazon involviert war? Amazon ist nämlich Eigentümer der Annapurna Labs aus Israel. Und alles dies sind nur hochkarätige Fusionen und Akquisitionen im Wert von mehreren hundert Millionen bis zu zehn Milliarden Euro, es haben auch viele kleinere Übernahmen stattgefunden. Kleine Unternehmen fusionieren zu größeren Einheiten, um dann, wenn sie zu groß werden, wieder versuchen, an Gewicht zu verlieren. NXP wurde von Philips abgespalten, Freescale war früher Teil von Motorola; HP teilte sich selbst und am Ende des Jahres 2016 kündigte Samsung ähnliche Pläne an. Es ist wie das Zusammenstoßen und Auseinanderbrechen von Planeten, Sternen und Universen. Gibt es vielleicht irgendwo ein schwarzes Loch, das die gesamte Branche aufsaugen möchte?

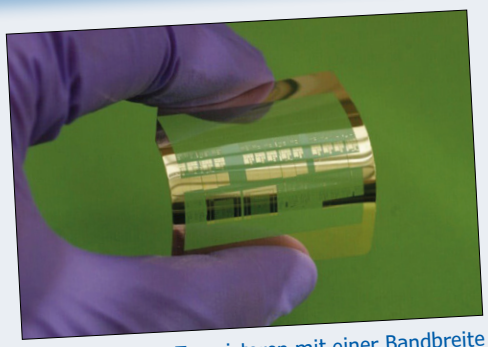
Color	1st band	2nd band	3rd band (multiplier)	4th band (tolerance)	Temperature Coefficient
Black	0	0	$x10^0$		
Brown	1	1	$x10^1$	±1%(F)	100 ppm
Red	2	2	$x10^2$	±2%(G)	50 ppm
Orange	3	3	$x10^3$		15ppm
Yellow	4	4	$x10^4$		25ppm
Green	5	5	$x10^5$	±0.5%(D)	
Blue	6	6	$x10^6$	±0.25%(C)	
Violet	7	7	$x10^7$	±0.1%(B)	
Gray	8	8	$x10^8$	±0.05%(A)	
White	9	9	$x10^9$		
Gold			$x10^{-1}$	±5%(J)	
Silver			$x10^{-2}$	±10%(K)	
None				±20%(M)	

RS279?

Die meisten Elektor-Leser kennen RS232, viele RS485, einige RS422, aber wer kennt RS279? Sie! Dies mag Sie überraschen. Schon 1952 gab es eine IEC-Norm für die Farbkodierung elektronischer Bauteile, aber seit 1963 ist RS279 die offizielle EIA-Norm für die Widerstands-Farbkodierung. SMDs besitzen keine Farbkodierung, sondern Buchstaben und Ziffern. Die aktuelle internationale Norm, die die Kennzeichnung von Widerständen und Kondensatoren definiert, heißt IEC 60062:2016.

Der Platinendrucker – noch eine nutzlose Maschine?

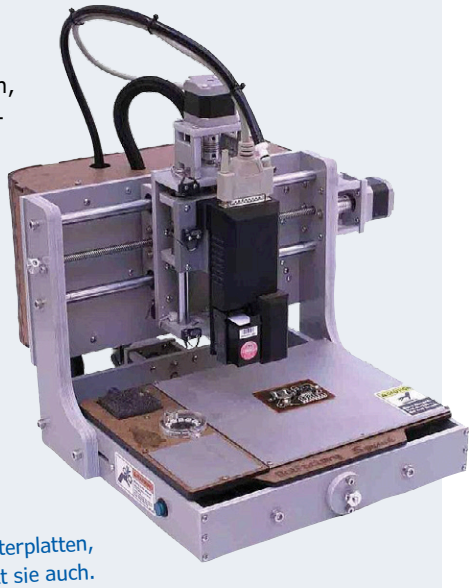
Der Desktop-Platinendrucker/Bestücker Voltera vereint Design und Funktionalität.



Diese kleinen Transistoren mit einer Bandbreite von 38 GHz wurden graviert und auf ein flexibles Substrat gedruckt. (Foto mit freundlicher Genehmigung von Jung-Hun Seo).

Wussten Sie, dass die Elektronik nicht dort wäre, wo sie heute ist, ohne eine vor etwa 500.000 Jahren entwickelte Technik? Ich meine nicht die Erfindung des Siliziums (dies geschah noch ein paar Milliarden Jahre früher), sondern die Gravur, „... eine spanende Bearbeitung, die lokal Material vom Werkstück abträgt“ (Danke, Wiki). Auch Leiterplatten oder Platinen können durch chemische, mechanische oder optische Gravurtechniken hergestellt werden, integrierte Schaltungen werden auf Siliziumwafern graviert. Auch das Drucken, eine mehr als 2000 Jahre alte Erfindung, hat sich als ziemlich wichtig für die Elektronik-Industrie erwiesen. Klar, ohne diese Technik könnten Sie diesen Text nicht lesen, aber vor allem beruht die Platinenfertigung darauf. Drucker sind heute in der Lage, Leiterplatten ohne Gravieren zu produzieren, indem sie leitende Tinte auf eine harte, meist ebene Oberfläche spritzen. Es werden auch Drucker entwickelt, die elektronische Schaltungen anstelle von Leiterplatten drucken. Es wird daran gearbeitet, für intelligente Verpackungen oder intelligente Kleidung oder ein anderes kluges Dingsbums einen einfachen 4-Bit-Mikrocontroller direkt auf Karton zu drucken. Heutzutage mag man sich fragen, was zum Teufel man mit einem 4-Bit-Prozessor anfangen soll, aber 1971, als Intel den 4-Bit-Prozessor 4004 auf den Markt brachte, war es eine Revolution, die Kampfflugzeuge in der Luft hielt.

Was würden Sie lieber in Ihrem Homelab haben? Einen Drucker, der gedruckte Leiterplatten druckt, die Sie selbst bestücken müssen, oder einen, der funktionsfähige Schaltungen ausspuckt? Vielleicht sollten wir einfach den Leiterplattendrucker ganz auslassen und uns auf den Elektronikdrucker konzentrieren?



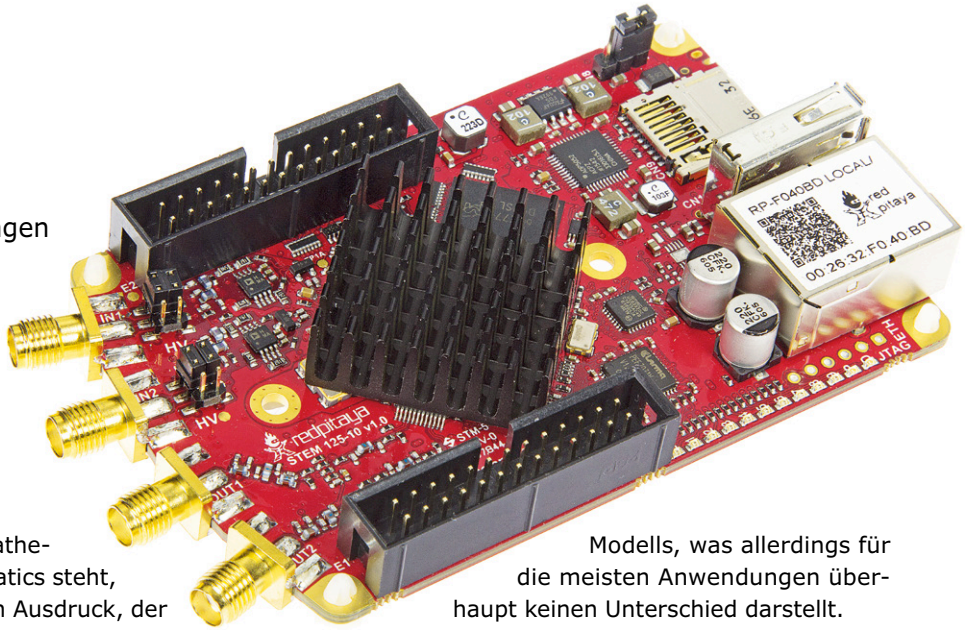
Squink von BotFactory druckt nicht nur Leiterplatten, sondern bestückt sie auch.

Muss ich haben!

Zum Herstellen von Gitarrenplektra oder zum (möglichst) zerstörungsfreien Prüfen maker-unfreundlicher Gadgets? Holen Sie sich diese Plektrum-Stanze!

Möchten Sie mitmachen? Bitte senden Sie Ihre Kommentare, Anregungen, Tipps und Tricks an labs@elektor.com!

Review:
STEMlab125-10
Der kleine Bruder des Red Pitaya



Von **Harry Baggen** (Elektor-Labor)

Das Mess- und Entwicklungssystem Red Pitaya ist seit Kurzem in zwei Ausführungen erhältlich, nämlich in einer 10-bit- und einer 14-bit-Version. Wie wollen hier den neuen „kleinen“ Red Pitaya vorstellen, der auf den Produktnamen STEMlab125-10 hört.

Red Pitaya ist unter Elektronikern mittlerweile ein bekannter Begriff. Die kleine rote Platine stellt sowohl ein leistungsfähiges Messgerät mit integriertem Generator als auch eine vielseitige FPGA-Entwicklungsplattform für eigene Applikationen dar. In Elektor haben wir schon zahlreiche Artikel über Anwendungen des Red Pitaya veröffentlicht und es ist sogar ein Buch in unserem Verlag erschienen [1].
Die Techniker von Red Pitaya haben ihr Produktangebot nun erneuert und erweitert. Zunächst wurden die Bezeichnungen geändert, alle verfügbaren Boards heißen nun „STEMlab“ (wobei STEM für Science, Technology, Engineering and Mathematics steht, ein Ausdruck, der in Lehreinrichtungen für technische und exakte Wissenschaften verwendet wird). So heißt der „alte“ Red Pitaya nun „STEMlab 125-14“, der nun ein junges und etwas abgespecktes Brüderchen namens STEMlab 125-10 erhält. Der Hauptunterschied, man ahnt es, liegt in der Auflösung der AD- und DA-Wandler. Statt 14 Bit sind es beim neuen Modell „nur“ 10 Bit, was sich auch im Verkaufspreis bemerkbar macht. Einen Nebenunterschied gibt es auch, er liegt im kleineren RAM des neuen Modells, was allerdings für die meisten Anwendungen überhaupt keinen Unterschied darstellt.

Weniger Bits sind mehr als genug!

Ich habe eine Zeit lang mit einem Prototyp des neuen Modells STEMlab 125-10 gespielt und ein Unterschied ist mir ehrlich gesagt überhaupt nicht aufgefallen. Beide Modelle verwenden nämlich die gleichen Apps und die Messergebnisse sehen auch identisch aus. Äußerlich kann man doch einen Unterschied bemerken: Die 10er-Version besitzt einige Verbinder weniger als die 14er-Version.

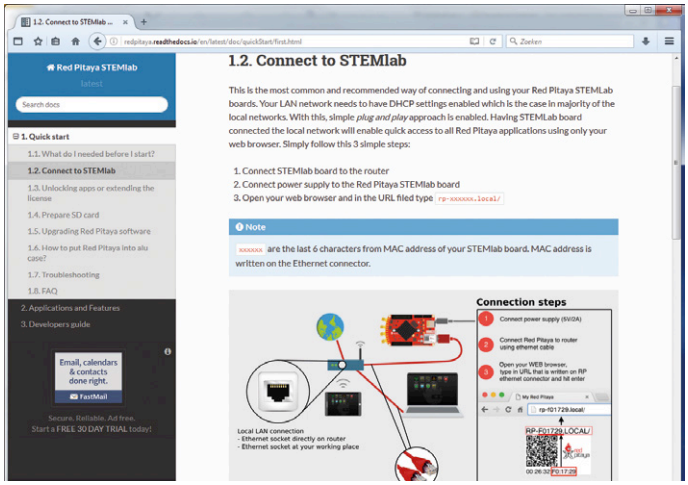


Bild 1. Die Dokumentation auf der Website von Red Pitaya.

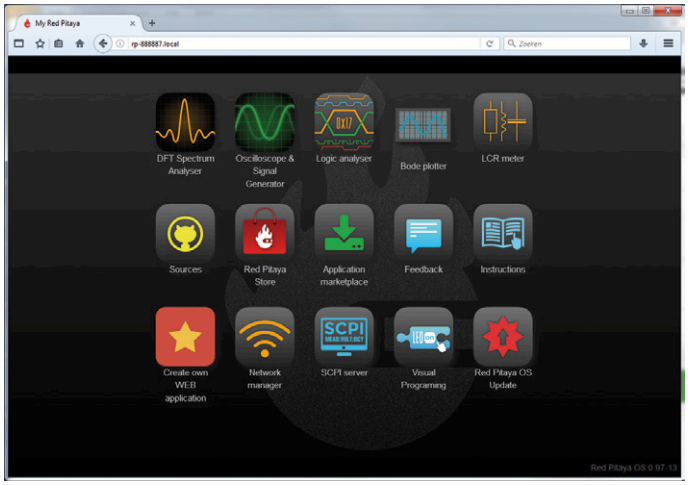


Bild 2. Das neue Benutzerinterface mit Icons à la Smartphone.

10 Bit, das erscheint doch recht mager im Vergleich zu den satten 14 Bit des großen Red Pitayas, doch wenn man bedenkt, dass die meisten Labortisch- und USB-Oszilloskope mit 8-Bit-Wandlern arbeiten, liefern 2 Bit mehr vier Mal genauere Ergebnisse! Im Oszilloskopbetrieb reichen 10 Bit für eine sehr detaillierte Darstellung eines Signals auf dem Monitor völlig aus. 14 Bit sind vor allem für diejenigen interessant, die eigene Applikationen entwickeln und dafür genaueste Messergebnisse benötigen.
Ein schöner Vorteil der neuen STEMlab-Reihe ist es, dass nun ein passendes Steckernetzteil und eine vorprogrammierte microSD-Karte zum Lieferumfang gehören. Bei der alten Version musste man zuerst das Image herunterladen und auf eine SD-Karte schreiben. Die neuen STEMlabs kann man dagegen auspacken und sofort damit loslegen, denn auf der microSD-Karte stehen auch die am häufigsten genutzten Programme wie Oszilloskop/Funktionsgenerator, Spektrum-, Logic- und Bodeanalyzer (bei meinem Prototyp noch nicht ganz einsatzbereit) sowie ein LCR-Meter direkt zur Verfügung. Man muss die Apps nicht mehr einzeln herunterladen und installieren. Übrigens: Die Programme für Oszilloskop/Funktionsgenerator und Spektrum-Analyzer sind die ehemaligen Pro-Versionen, die anderen Apps sind in ihrem Layout und ihrer Bedienung angeglichen.

Ein einfacher Start

Wenn man zum ersten Mal ein solches STEMlab vor sich auf dem Labortisch hat, sollte man (doch, ja!) erst einmal die Dokumentation (**Bild 1**) lesen, die sich hinter dem Hilfe-Knopf auf der Red-Pitaya-Webseite verbirgt. Glücklicherweise ist diese Doku ein Stück übersichtlicher im Vergleich zur alten geworden und nach ein paar Seite Lektüre kann man schon, weil es in den Fingern juckt, das Gerätchen anschließen und damit loslegen. Dafür schließt man das Ding am besten direkt mit einem Ethernet-Kabel am Router an. Die Erkennung des neuen Boards im Netzwerk verläuft auch viel bequemer als beim Vorgängermodell: Man muss im Browser nur einige Befehle und die Ziffern der MAC-Adresse eintippen, dann erscheint das User-Interface mit einer Reihe von Smartphone-artigen Icons für die (vor-)installierten Programme, aber auch für den Zugang zum Marketplace für den Download weiterer Programme oder für eine OS-Update-Funktion aus dem Browser heraus. Möchte man eine WLAN-Verbindung verwenden, kann man das STEMlab über den Netzwerk-Manager beim WLAN-Netzwerk anmelden und ein Passwort eintragen, woraufhin eine IP-Adresse erscheint, unter der das Gerät dann im Netzwerk erreichbar ist.
An den Apps hat sich, soweit ich es sehen konnte, wenig oder nichts verändert. Der Artikel „Neue Apps für Red Pitaya“ [3] beschriebt das Oszilloskop/Funktionsgenerator-Programm und den Spektrum-Analyzer. Dabei gibt es keinen auf dem Monitor sichtbaren Unterschied zwischen dem 10-bit- oder dem 14-bit-Modell. Beide Apps sind in der Dokumentation ausführlich beschrieben.

Fazit

Mit dem STEMlab 125-10 gibt es ein kleineres und preisgünstigeres Red-Pitaya-Modell, das sich beim Gros der Anwendungen nicht hinter seinem großen Bruder verstecken muss. Dank der mit den meistgebrauchten Apps vorprogrammierten SD-Karte kann der Anwender das STEMlab sofort in Betrieb nehmen - out of the box.
Es gibt bei Red Pitaya verschiedene Pakete mit dem STEMlab 125-10 und dem STEMlab 125-14, angefangen bei der nackten Platine ohne Zubehör (nur das Steckernetzteil und die microSD-Karte gehören zu jedem Paket) bis zum reichhaltigen Kit mit allem Zubehör, das man aber auch einzeln kaufen kann: Gehäuse, LCR-Modul, ein Satz Messkabel und vieles mehr.

Für die meisten Elektroniker, die noch keinen Red Pitaya ihr eigen nennen, ist das Diagnostic-Kit mit dem STEMlab 125-10 das wohl interessanteste Paket. Es enthält neben dem STEMlab 125-10 ein Steckernetzteil, die vorprogrammierte microSD-Karte, zwei Oszilloskop-Messköpfe, einen WLAN-Dongle, zwei SMA/BNC-Adapter, zwei 50-Ω-Abschlussstecker, zwei SMA-T-Adapter, neun Messclips mit einem Anschlusskabel für den Logic-Analyzer, ein Ethernet-Kabel und ein Klarsicht-Kunststoffgehäuse. Damit holt man sich einen vollständigen Messplatz ins Haus! ◀

(160366)

Weblinks

[1] www.elektor.de/red-pitaya-for-test-and-measurement

[2] www.redpitaya.com

[3] www.elektormagazine.de/150595

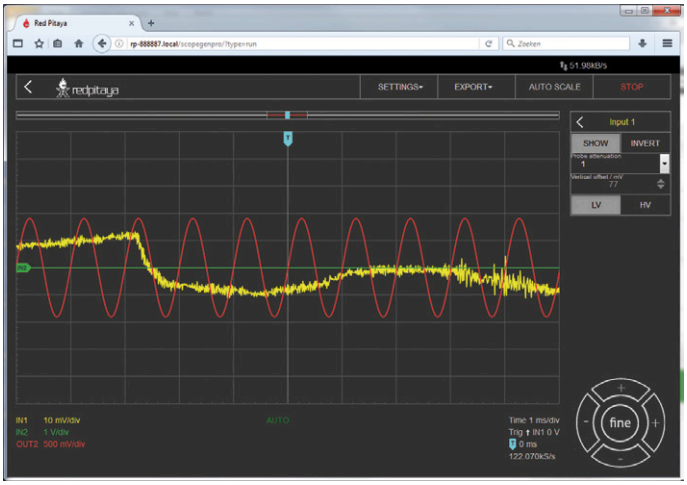


Bild 3. Das Oszilloskop-Programm im Lieferumfang ist identisch mit der ehemaligen Pro-Version.

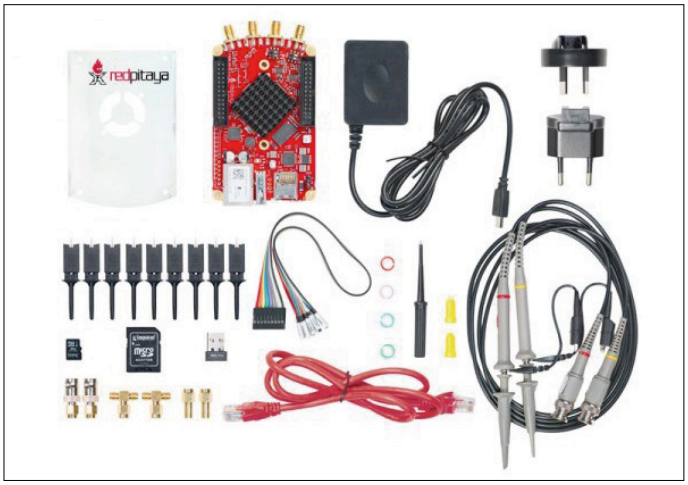


Bild 4. Mit dem Diagnostic-Kit holt man sich einen kompletten Messplatz ins Labor.

Zufallszahlen-Generator

Erfüllt jeden NIST-Test

Als ich mit einem Software-Zufallszahlengenerator gearbeitet habe, ist mir aufgefallen, dass Zahlen, die aus früheren Werten berechnet werden, eigentlich gar nicht so willkürlich sind wie sie sein sollten. Ich habe dann darüber nachgedacht, wie ein mit Standard-Bauteilen aufgebautes elektronisches Gerät wirklich echte Zufallszahlen erzeugen kann.

INFOS ZUM PROJEKT

Computer & Internet
Sicherheit | Kryptografie
Gaming

Einsteiger
Fortgeschrittener
➡ Experte

ungefähr 4 Stunden

SMD-Löten
AVR-Programmer
einstellbares Labornetzgerät

etwa 150 €

Von **Luka Matić** (Kroatien)

Als ich mich mit den Grundlagen der Kryptographie beschäftigte, entdeckte ich, warum ein guter Zufallszahlengenerator (Random Number Generator RNG) eine sehr nützliche Sache ist. Sichere Datenverschlüsselung zum Beispiel ist ohne ihn nahezu unmöglich. Gaming- und Glücksspielanwendungen erfordern ebenfalls hochklassige RNGs. Und wissen Sie, dass Sie die Zahl Pi anhand von Zufallszahlen ableiten können? Suchen Sie mal im Internet nach „Buffon“ und „Pi“!

Hochwertige Zufallssignalgeneratoren sind zwar im Handel erhältlich, neigen jedoch dazu, viel zu viel Geld zu kosten. Deshalb habe ich beschlossen, selbst einen RNG aus billigen und leicht erhältlichen Bauteilen zu konstruieren.

Die Schaltung

Die Zufälligkeit von verschiedenen elektronischen Rausch-Typen (shot, thermal, flicker, popcorn, avalanche, ...) ist ja bekannt. Deshalb verwendet mein RNG das Avalanche-Rauschen von Zenerdioden als Rauschquelle, die im Vergleich zu anderen eine hohe Bandbreite und viel höhere Amplitude aufweist. In der Schaltung in **Bild 1** erzeugen die beiden 12-V-Z-Dioden D7 und D10 zwei Rauschsignale. Ein Differenzverstärker mit T4, T5 und T6 verstärkt die Differenz beider Signale, um die Wahrscheinlichkeit zu verbessern, dass das Rauschsignal auch wirklich zufällig ist. Dieser Verstärker entfernt gleichzeitig auch Gleichtaktstörungen, die beide Dioden beeinflussen könnten.

Die nächste Stufe (T7) verstärkt das Rauschsignal weiter bis zu 0,5...1 V_{pp}, was für den Analogkomparator des ATtiny2313 ausreichend hoch ist. Sie

fungiert auch als einstellbares Korrekturfilter, das es dem RNG ermöglicht, die Qualitätskriterien für Zufallssignale zu erfüllen, die von internationalen Normierungsorganisationen vorgegeben werden (siehe **Kasten**).

Der Ausgang des analogen Komparators wird mit einer Rate von etwa 800 kHz abgetastet. Bestimmte Muster im Zufallsbitstrom können so mit den Subharmonischen der Abtastfrequenz korreliert werden, die in dem Rauschsignal vorhanden sind. Die Bitmuster 0x00 und 0xFF entsprechen der Subharmonischen von 50 kHz, 0x0F und 0xF0 entsprechen 100 kHz, 0x33 und 0xCC korrespondieren mit 200 kHz und schließlich die Muster 0x55 und 0xAA mit 400 kHz. Tritt ein

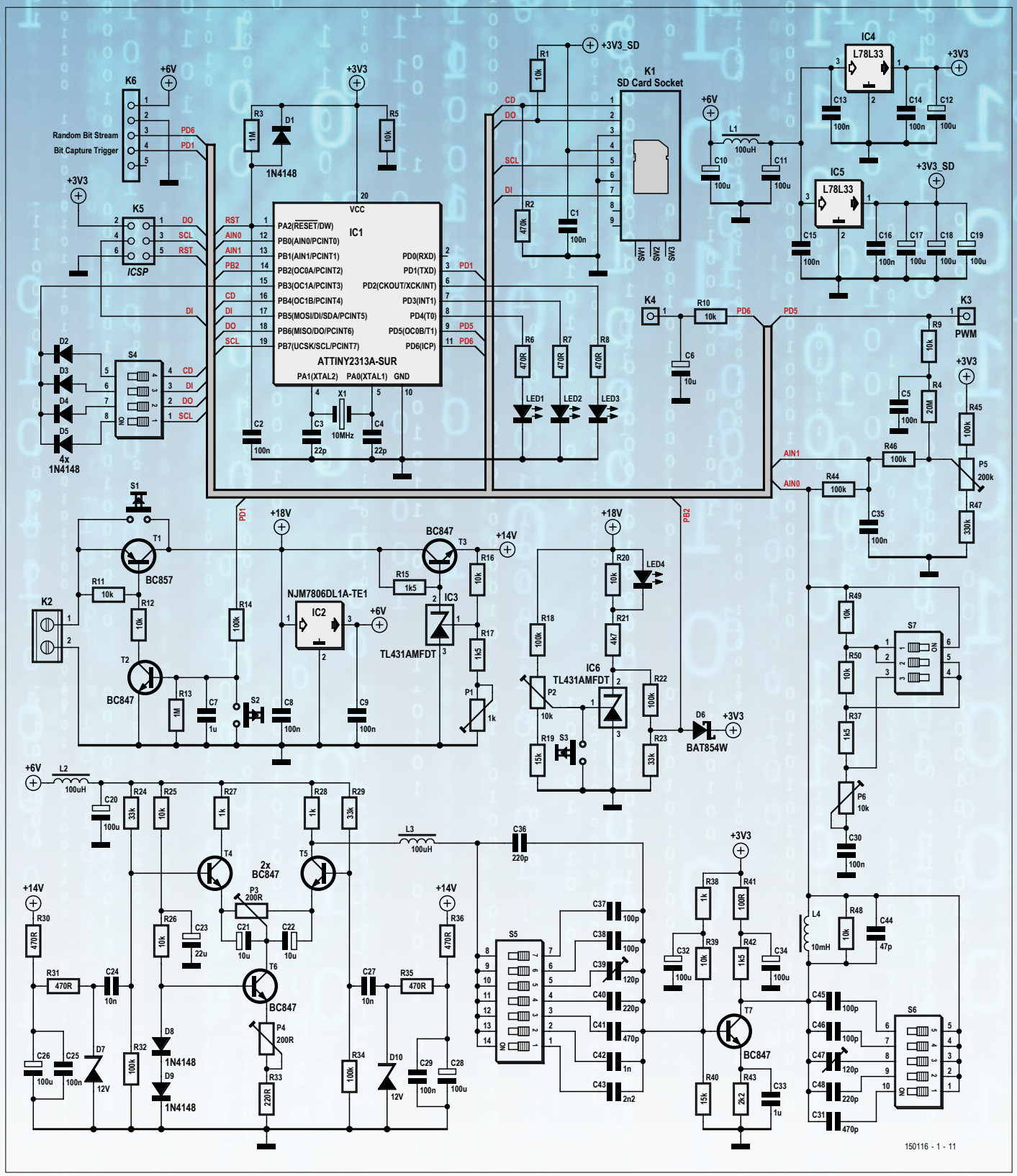


Bild 1. Der Zufallszahlen-Generator RNG vereinigt auf eine clevere Art und Weise analoge und digitale Elektronik, um einen echten Zufallsbitstrom mit hoher Qualität zu erzeugen.

Tabelle 1. Die endgültige Dateigröße ergibt sich aus der Addition aller aktiven Schalter plus 16 MB. Wenn alle Schalter auf OFF stehen, beträgt die Dateigröße demnach 16 MB.

Schalter	Dateigröße
S4-1	2048 MB
S4-2	1024 MB
S4-3	512 MB
S4-4	256 MB

bestimmtes Bitmuster zu oft oder zu selten auf, deutet das darauf hin, dass der Frequenzgang des Korrekturfilters geändert werden muss, um die Verstärkung bei der auffälligen Frequenz zu verringern oder zu erhöhen. Dies kann mit den DIP-Schaltern S5, S6 und S7, den Trimmkondensatoren C39, C47 und Trimmpoti P6 geschehen.

Das abgetastete Ausgangssignal des Analogkomparators wird auf der SD-Karte in einer großen Datei aufgezeichnet. Die SD-Karte muss dazu FAT32-formatiert sein (siehe **Kasten**). Die gewünschte Dateigröße wird an DIP-Schalter S4 von 16 MB bis 3,8 GB eingestellt (siehe **Tabelle 1**). Nach der Aufzeichnung wird eine Datei „random.hex“ auf der SD-Karte erstellt.

Der zufällige Bitstrom wird auch auf PD6 (an K4) ausgegeben, so dass seine

STÜCKLISTE RNG

Widerstände:

alle 0805, 100 mW, 5%
R1,R5,R9,R10,R11,R12,R16,R20,R25,R26,R39, R48,R49,R50 = 10 k
R2 = 470 k
R3,R13 = 1 M
R4 = 20 M
R6,R7,R8,R30,R31,R35,R36 = 470 Ω
R14,R18,R22,R32,R34,R44,R45,R46 = 100 k
R15,R17,R37,R42 = 1k5
R19,R40 = 15 k
R21 = 4k7
R23,R24,R29 = 33 k
R27,R28,R38 = 1 k
R33 = 220 Ω
R41 = 100 Ω
R43 = 2k2
R47 = 330 k
P1 = Trimmpoti, 1 k, 11 Gänge, SMD
P2,P6 = Trimmpoti, 10 k, 11 Gänge, SMD
P3,P4 = Trimmpoti, 200 Ω, 11 Gänge, SMD
P5 = Trimmpoti, 200 k, 11 Gänge, SMD

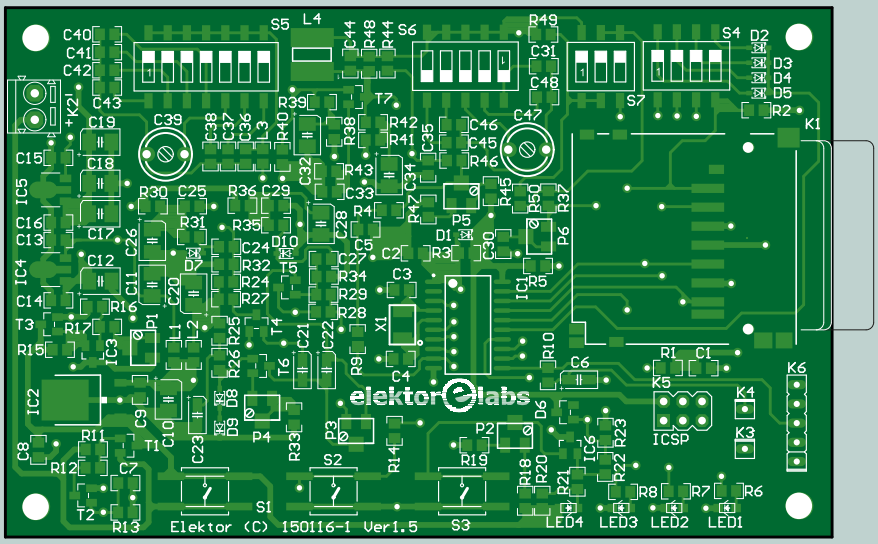
Kondensatoren:

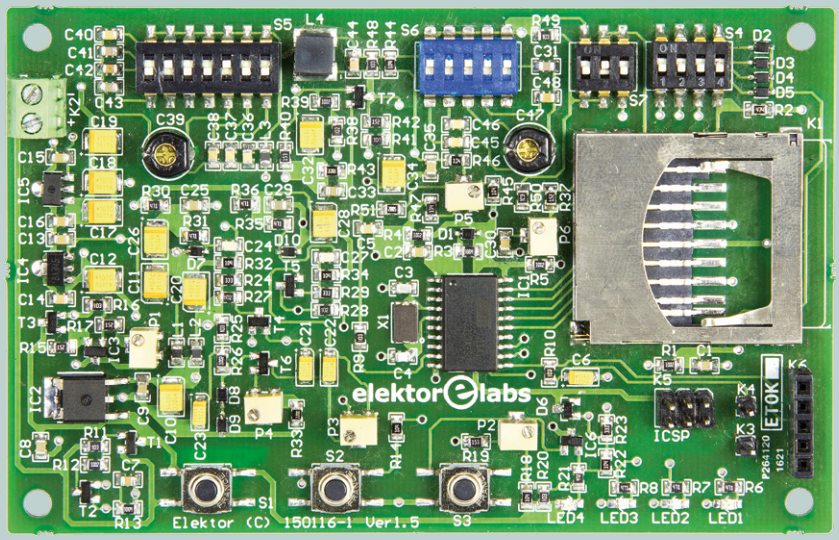
alle 0805
C1,C2,C5,C8,C9,C13,C14,C15,C16,C25,C29, C30,C35 = 100 n
C3,C4 = 22 p
C6,C21,C22 = Elko 10 µ, 6V3, Gehäuse A
C7,C33 = 1 µ
C10,C11,C12,C17,C18,C19,C20,C26,C28, C32,C34 = Elko 100 µ, 10 V, Gehäuse B
C23 = Elko 22 µF, 10 V, Gehäuse A
C24,C27 = 10 n
C31,C41 = 470 p
C36,C40,C48 = 220 p
C37,C38,C45,C46 = 100 p
C39,C47 = Trimmkondensator 120 p (eBay)
C42 = 1 n
C43 = 2n2
C44 = 47 p

Induktivitäten:

L1,L2,L3 = 100 µ, 0805
L4 = 10 m, 2220

Halbleiter:
IC1 = ATtiny2313V-10SUR, programmiert
IC2 = L7806CD2T-TR (oder NJM7806DL1A-TE1)





IC3,IC6 = TL431AMFDT
IC4,IC5 = L78L33ACUTR
D1,D2,D3,D4,D5,D8,D9 = 1N4148WS
D6 = BAT854W
D7,D10 = BZX384-C12
T1 = BC857CW
T2,T3,T4,T5,T6,T7 = BC847BW
LED1,LED4 = grün, 2 x 1,25 mm
LED2 = rot, 2 x 1,25 mm
LED3 = gelb, 2 x 1,25 mm

Außerdem:

X1 = Quarz 10 MHz, 18 p, 5,0 x 3,2 mm
S1,S2,S3 = Taster
S4 = DIP-Schalter, 4-fach
S5 = DIP-Schalter, 7-fach
S6 = DIP-Schalter, 5-fach
S7 = DIP-Schalter, 3-fach
K1 = SD-Kartenaufnahme SDBMF-00915B0T2
K2 = 2-polige Platinenanschlussklemme, Raster 3,5 mm
K3,K4 = 1 Stift
K5 = 2x3-polige Stiftleiste, Raster 0,1“
K6 = 1x5-polige Stiftleiste, Raster 0,1“

Qualität in Echtzeit von einem Bitstromanalysator (BSA) überwacht werden kann. Dabei handelt es sich um ein Werkzeug, das die Häufigkeit des Auftretens bestimmter Bitmuster erfasst - sehr praktisch für die Einstellung des Filters um T7. Um eine Synchronisierung mit dem Bitstrom zu ermöglichen, erzeugt PD1 am Anfang eines jeden Bits einen kurzen (<1 µs) low-aktiven Impuls. Diese Impulse sind so kurz, dass sie den RNG nicht beeinflussen können.

Spannungsversorgung

Um all die Störungen zu umgehen, die aus einer Netzversorgung herrühren könnten, wird das Gerät aus zwei in Reihe geschalteten 9-V-Blöcken batterieversorgt. Diese hohe Spannung von 18 V ist wegen der beiden 12-V-Z-Dioden D7 und D10 erforderlich.

Die Transistoren T1 und T2 stellen zusammen mit dem Ausgang PD1 ein Flipflop zum Ein- und Ausschalten dar. Drückt man die Taste S1, wird der Stromkreis eingeschaltet, so dass der Controller zum Leben erweckt wird. Port PD1 wird aktiviert und hält damit den Zustand stabil, auch wenn man S1 wieder loslässt.

Die Schaltung mit IC6 arbeitet mit der ungeregelten Batteriespannung als Low-Bat-Anzeige. Über PB2 wird der Controller informiert, wenn die Spannung auf etwa 15,4 V absinkt. Der Controller beendet daraufhin das Beschreiben der SD-Karte und schaltet danach die Betriebsspannung über PD1 ab. Damit werden sowohl eine Verfälschung der Daten wie auch ein übermäßiges Aussaugen der Batterien (Akkus) verhindert. Mit S3 kann man eine leere Batterie simulieren: Ein Druck auf den Taster bricht jederzeit den Betrieb des Generators unter Beibehaltung der Daten ab.

Kann man S3 als Soft-stop-Taste bezeichnen, so ist S2 ist dagegen eine Hard-stop-Taste, die das Gerät sofort ausschaltet. Wenn Sie diese Taste drücken, während eine Aufzeichnung läuft, wird die Datei höchstwahrscheinlich korrumpiert sein. Verwenden Sie diese Taste deshalb nur nach dem Einstellen von Trimmpoti P5 oder wenn etwas völlig daneben läuft.

Die LEDs

Wenn Sie S1 drücken, um den Generator einzuschalten (und die Batteriespannung hoch genug ist), leuchtet LED4 auf. Postwendend folgt LED3 ihrem Beispiel, dann auch LED1, um zu zeigen, dass die SD-Karte korrekt initialisiert wurde. LED1 verlischt dann wieder und die Aufzeichnung der Zufallsbits beginnt. Leuchtet LED1 wieder auf, so bedeutet dies, dass die Zufallszahlendatei in der gewünschten Länge erfolgreich auf die SD-Karte geschrieben wurde. LED1 bleibt auch hell, solange die Batteriespannung hoch genug ist. LED3 blinkt während der Aufzeichnung: kurz am Ende jedes Clusters und lang bei der Aktualisierung der FAT-Tabelle. Nicht schön ist es, wenn LED2 aufleuchtet: Irgendein Problem ist während der SD-Kartenkommunikation aufgetreten!

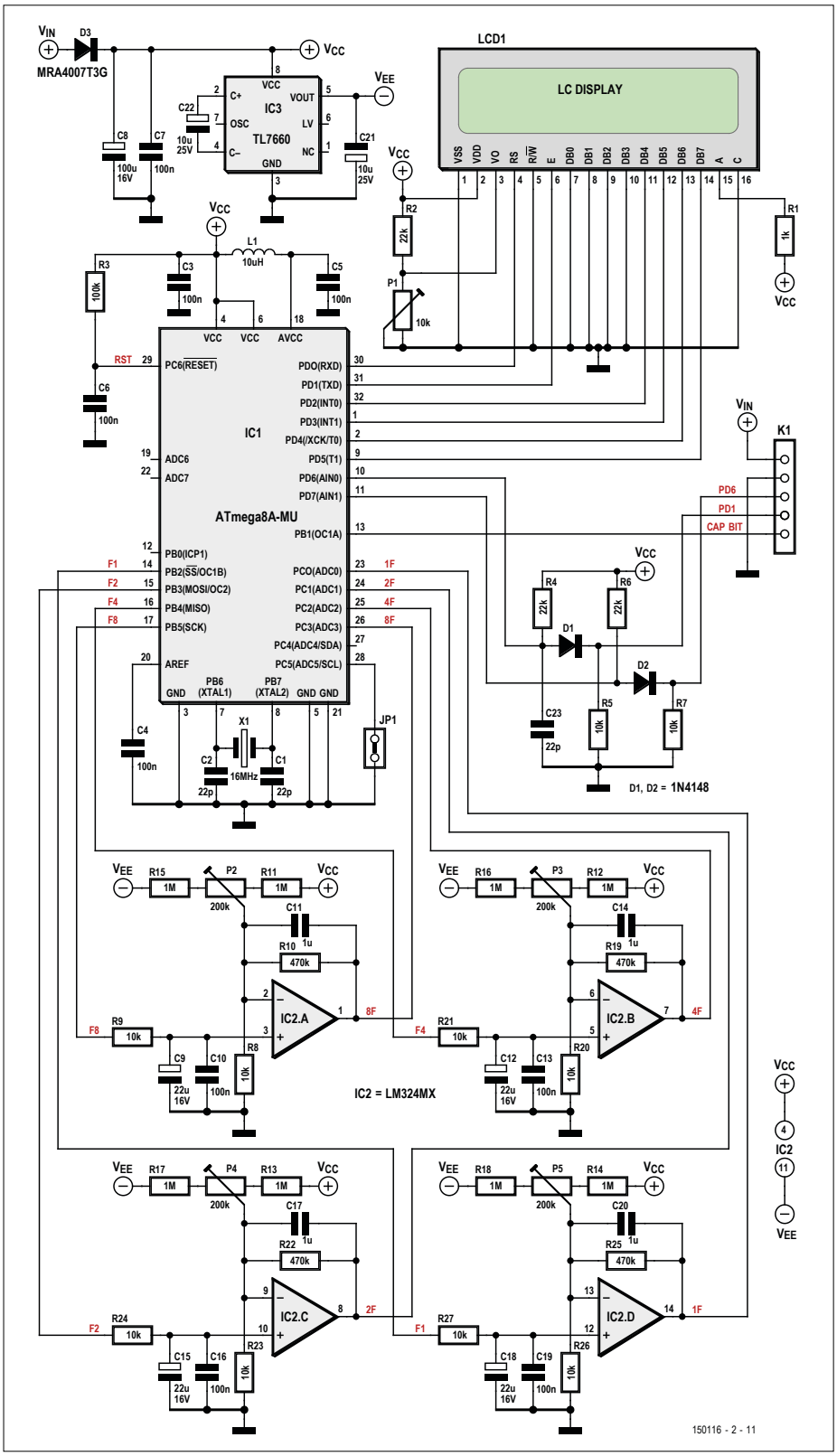


Bild 2. Der Bitstrom-Analysator BSA ermöglicht eine Abstimmung des RNG in Echtzeit.

Es geht noch besser...

Die Zufälligkeit der durch den RNG erzeugten Daten kann noch weiter verbessert werden, wenn zwei aufgezeichnete Sequenzen der gleichen Länge bitweise XOR-verknüpft werden. Dies entspricht der Verschlüsselung einer Sequenz mit einer anderen, auch One-Time-Pad-Verschlüsselung (OTP) genannt. Die resultierende Sequenz ist viel flacher, es treten Abweichungen von lediglich ±2-3 % auf, was manchen NIST-Tests für viel längere Sequenzen genügt. So läuft der Runs-Test für Sequenzen länger als 500.000 Bits, im Gegensatz zu den 50.000 Bits, die ohne Nachbearbeitung erreicht werden können.

Bit-Stream-Analyzer

Zufallsdaten mit hoher Qualität zu erzeugen ist nicht trivial. Der Generator besitzt eine Reihe von korrekt einzustellenden Widerständen und Kondensatoren. Auch wenn es möglich wäre, den Bitstrom aufzuzeichnen, zu analysieren, den Generator neu einzustellen und diesen Vorgang so oft wie nötig zu wiederholen, ist dieses Vorgehen recht unpraktisch und langwierig. Aus diesem Grund habe ich ein spezielles Werkzeug entwickelt, den Bit-Stream-Analyzer (BSA). Damit werden die Dinge einfacher, weil die Effekte der Abstimmung in Echtzeit beobachtet werden können.

Der BSA besteht aus einem ATmega8 und einem alphanumerischen 2x16-Display (**Bild 2**), die Schaltung ist an K6 des Generators angeschlossen. Der Analyzer liest den vom Generator erzeugten Zufallsbitstrom an PD6 und das Bitsynchronisationssignal an PD1. Der BSA ist eine Art Spektrum-Analyzer, der auf die ersten vier Subharmonischen der Generator-Abtastfrequenz von 800 kHz abgestimmt ist. Es zeigt die Amplituden von f1 (50 kHz), f2 (100 kHz), f4 (200 kHz) und f8 (400 kHz) sowie das 0/1-Verhältnis an. Der Controller wird von einer Art „Analogcomputer“ in Form von vier Operationsverstärkern (IC2) unterstützt, der die Pegel der vier Subharmonischen berechnet. Der ATmega8 ist zu langsam dafür und kümmert sich deshalb nur um das 0/1-Verhältnis und Ein- und Ausgangssignal der Opamps: Für jedes Auftreten eines charakteristischen Musters 0x00, 0x0F, 0x33 oder 0x55 gibt der Controller einen kurzen Impuls mit konstanter Länge auf PB2...PB5 aus und liest den integrierten Analogwert an PC0...PC3 wieder ein.

Rechts oben im Display ist das 0/1-Verhältnis zu sehen, das etwa 512 betragen sollte. Es wird nicht vom BSA, sondern von P5 des Generators und dem PWM-Auto-Offset gesteuert. Die vier Subharmonischen sollten die gleiche Größenordnung aufweisen und innerhalb von etwa 10 % liegen, die genauen Werte sind aber nicht so wichtig.

Bevor der Analysator eingesetzt werden kann, ist eine DC-Offsetkorrektur der Opamps erforderlich:

1. Verbinden Sie mit Jumper JP1 PC5 und GND. Dann wird der BSA eingeschaltet oder zurückgesetzt. Der Controller startet in den Zero-Offset-Einstellmodus, was durch das Zeichen „Z“ im Display angezeigt wird.
2. Die Trimmpotis P2...P5 so drehen, dass die Werte von f1, f2, f4 und f8 geringfügig über Null (zwischen 10 und 20) liegen.

STÜCKLISTE BSA

Widerstände:

alle 0805, 100 mW, 5 %
R1 = 1 k
R2,R4,R6 = 22 k
R3 = 100 kΩ
R5,R7,R8,R9,R20,R21,R23,R24,R26,R27 = 10 k
R10,R19,R22,R25 = 470 k
R11,R12,R13,R14,R15,R16,R17,R18 = 1 M
P1 = Trimmpoti 10 k, SMD
P2,P3,P4,P5 = Trimmpoti 200 k

Kondensatoren:

alle 0805
C1, C2,C23 = 22 p
C3,C4,C5,C6,C7,C10,C13,C16,C19 = 100 n
C8 = Elko 100 µ, 16 V, radial, SMD
C9,C12,C15,C18 = Elko 22 µ, 16 V, radial, SMD
C11,C14,C17,C20 = 1 µ
C21,C22 = Elko 10 µ, 25 V, radial, SMD

Induktivität:

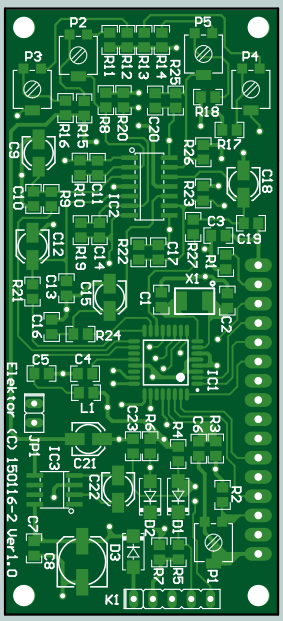
L1 = 10 µ, 0805

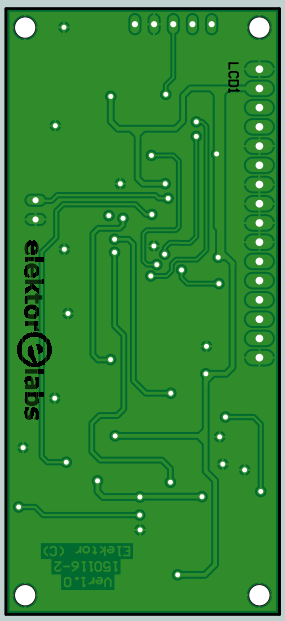
Halbleiter:

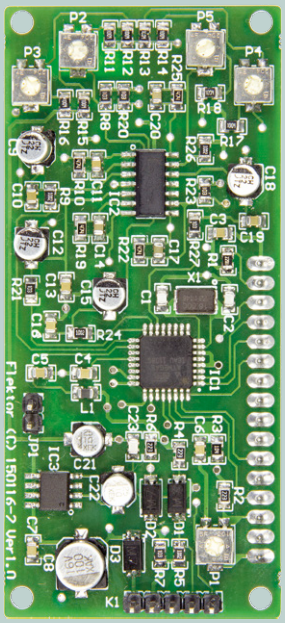
D1,D2 = 1N148WS
D3 = MRA4007T3G
IC1 = ATmega8-16AU
IC2 = LM324MX
IC3 = TL7660CD

Außerdem:

X1 = Quarz 16 MHz, 18 p, 5x3,2 mm
JP1 = 2-polige Stiftleiste, Raster 0,1“
Jumper, Raster 0,1“
K1 = 5-polige Buchsenleiste, Raster 0,1“ (Lötseite)
LCD1 = Alphanumerisches Display, 2x16
1x16-polige Stiftleiste (für LCD), Raster 0,1“
1x16-polige Buchsenleiste (für LCD), Raster 0,1“







3. Entfernen Sie den Jumper JP1 und setzen Sie den BSA zurück (oder schalten Sie ihn aus/an).

Abgleich des Generators

Beim Bau jedes Präzisionsgeräts mit preiswerten Komponenten müssen normalerweise ein oder mehrere Parameter manuell angepasst werden. Beim RNG ist das nicht anders:

Trimmpoti P1

So drehen, dass 14 V am Emitter von T3 anliegen.

Trimmpoti P2

P2 bestimmt die Spannung, die die Lowbat-Warnung auslöst (LED4 verlischt). Wenn der RNG mit zwei 9-V-NiMh-Akkus betrieben wird, sollte diese Schwelle 15,4 V betragen (entspricht einer Entladung von 1,1 V je Zelle). Schließen Sie eine einstellbare Spannungsquelle an K2 an und fahren Sie die Spannung von 16 V langsam herunter. Stellen Sie P2 so ein, dass die LED4 dunkel wird, wenn die Spannung an K2 nur noch 15,4 V beträgt.

Trimmpotis P3 und P4

Die Kollektorströme der Transistoren T4 und T5 müssen 1 mA betragen. Stellen Sie P3 (Symmetrie) und P4 (Gesamtstrom) so ein, dass an R27 und R28 ein Spannungsabfall von je 1 V gemessen wird.

Trimmpoti P5

1. Schalten Sie das Gerät aus (drücken Sie die Hard-stop-Taste S2).
2. Halten Sie die Soft-stop-Taste S3 gedrückt und drücken Sie S1 (Start), um in den Testmodus zu gelangen. LED3 leuchtet auf.
3. Schließen Sie ein Oszilloskop oder ein DC-Voltmeter an PD5 an, um das

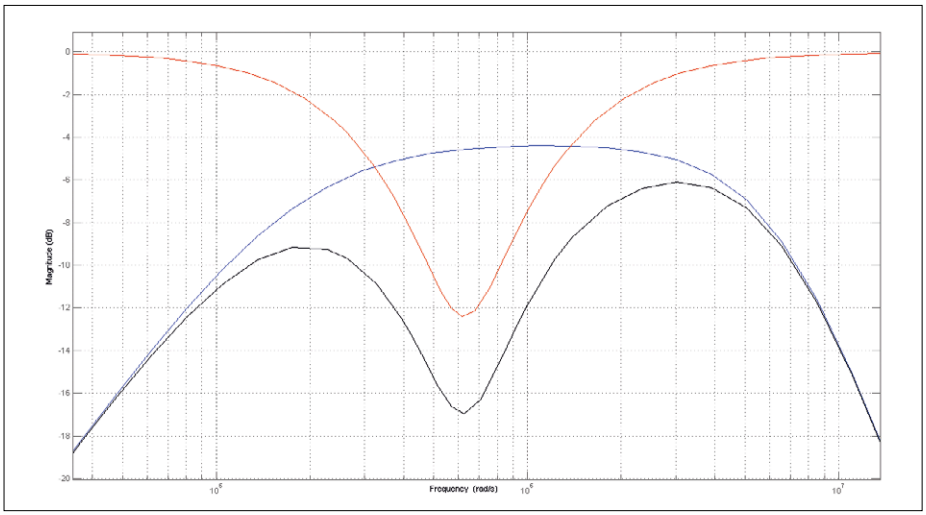


Bild 3. Die simulierte Transferfunktion des Filters mit T7 (schwarze Kurve). Die blaue Kurve zeigt die Transferfunktion von T7. Die untere Eckfrequenz wird mit S5 und den Kondensatoren C36... C43 eingestellt. Die rote Kurve schließlich beschreibt die Übertragungsfunktion des Korrekturfilters L4-R48-CS6-RS7. Die Transistorverstärkung wurde hier nicht berücksichtigt, die schwarze und die blaue Kurve liegen in Wirklichkeit um etwa 30 dB höher.

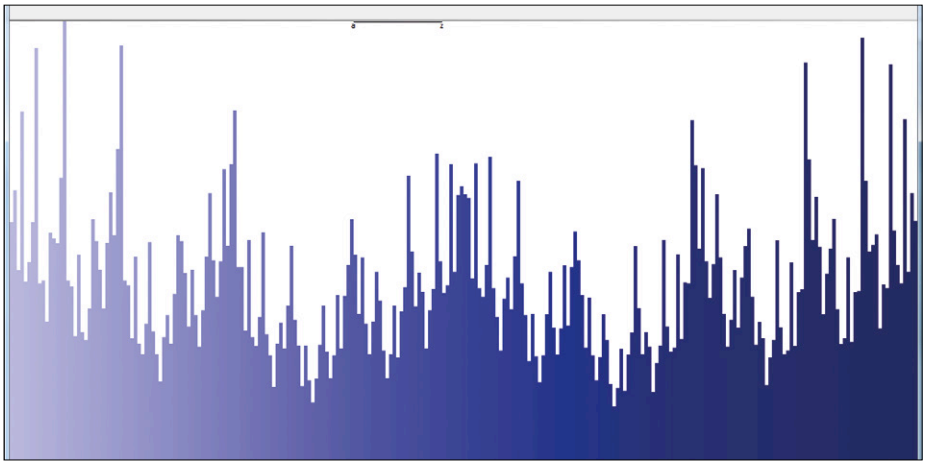


Bild 4. Dieses Histogramm der vom RNG erzeugten Daten zeigt ein schlecht abgestimmtes Korrekturfilter. Schaut man nur nach den Balken ganz rechts und ganz links (Auftreten von 0xFF und 0x00), scheint das 0/1-Verhältnis in Ordnung, aber wenn man die Häufigkeit von 0x0F und 0xF0 betrachtet, die dreimal höher ist als die von 0x55 und 0xAA, so offenbart sich die mangelhafte Abstimmung.

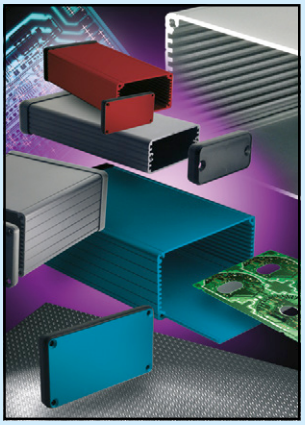
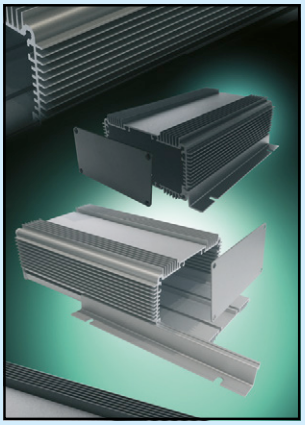
Anzeige

HAMMOND MANUFACTURING®

Profil - Gehäuse
Standard und Kühlkörper

www.hammondmfg.com/1455.htm
www.hammondmfg.com/1455NHD.htm

01256 812812
sales@hammondmfg.eu

Einige Feinheiten der SD-Karte

Der (Assembler-)Code für den ATtiny2313 funktioniert nur mit FAT32-formatierten SDHC-Karten. FAT16 oder SD-Karten, die nicht vom Typ SDHC oder größer als 4 GB sind (die 32-Bit-Pointer können nur 4 GB ansprechen) werden nicht unterstützt. Das ist nicht wirklich ein Problem, da FAT32 keine Dateien größer 4 GB erlaubt und das Schreiben solcher Dateien auch ohnehin zu lange dauern würde.
Ein FAT32-Treiber mit allen Raffinessen würde viel zu viel Speicherplatz des kleinen Controllers beanspruchen. Doch es müssen nur die grundlegenden FAT32-Funktionen (um eine einzelne Datei in das Root-Verzeichnis zu schreiben) implementiert werden. Wegen dieser Einfachheit und einiger Sicherheitsmaßnahmen zum sauberen Ausschalten kann die letztendliche etwas von der avisierten Größe abweichen.
Die SD-Karte sollte aus zwei Gründen mit einer Clustergröße von 8 KB formatiert werden:

- Je kleiner der Cluster, desto öfter wird die FAT-Tabelle aktualisiert (alle 2 s bei Sektoren von 1 KB). Das Signal zum Bitstromanalysator wird während der Aktualisierung der FAT-Tabelle gestoppt, so dass die Analysatorwerte abfallen, weil sich die Eingangskondensatoren entladen. Wenn dies zu oft passiert, wird die Einstellung des Generators aufgrund der schwankenden Messungen schwierig.
- Je größer der Cluster, desto länger ist das Intervall der FAT-Tabellen-Updates (mehr als 60 s für 32-KB-Sektoren). Die PWM-Auto-Offset-Korrektur findet bei jedem FAT-Tabellen-Update statt. Ist die Zeit zwischen den Aktualisierungen zu lang, wird die automatische Offsetkorrektur unwirksam.

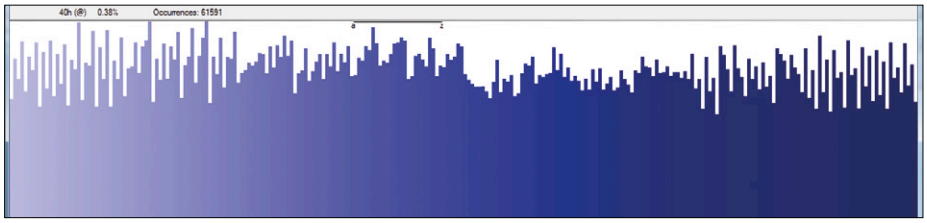


Bild 5. Nach einer sauberen Abstimmung des Korrekturfilters sieht das Histogramm schön flach aus. Damit sind Zufallszahlen der höchsten Qualität garantiert!

PWM-Signal zu überwachen, das die Gleichspannungsdrift kompensiert. Stellen Sie P5 so ein, dass sich das PWM-Signal bei etwa 50 % stabilisiert.

4. Der Testmodus dauert etwa 40 s, dann wird der PWM-Wert in das interne EEPROM der MCU geschrieben. Der Vorgang ist abgeschlossen, wenn die LED1 wieder leuchtet. Starten Sie den Testmodus erneut, wenn Sie mehr Zeit dafür benötigen. Wenn die PWM bei 0 % oder 100% „festhängt“, leuchtet LED2 und der PWM-Wert wird nicht in das EEPROM geschrieben. Dann müssen Sie den Testmodus erneut starten.

S5, S6 und S7

Bild 3 zeigt die Übertragungsfunktion der Schaltung um T7. S5 und die Kondensatoren bestimmen die untere Grenzfrequenz. Mit S6 wird die Resonanzfrequenz des Filters eingestellt:

$f_{res} = 1 / (2\pi \times \sqrt{(L4 \times C_{S6})})$ [Hz]

wobei S7 die maximale Tiefe festlegt:

$20 \times \log_{10}(R_{S7} / (R_{S7} + R48))$ [dB].

Auto-Abgleich

Der letzte PWM-Wert wird immer im internen EEPROM gespeichert. Im normalen Aufzeichnungsmodus wird das Tastverhältnis fortwährend eingestellt. Am Ende einer Aufzeichnung (bei Beendigung oder aufgrund eines Soft-Stops) wird das aktuelle PWM-Tastverhältnis in das EEPROM geschrieben, damit es beim nächsten Einschalten des RNG zur Verfügung steht.
Daher sollten Sie zunächst eine kleine 16-MB-Datei aufzeichnen (und verwerfen), so dass das PWM-Signal vom Controller eingestellt werden kann. Dann wird die gewünschte Dateigröße am Mäuseklavier S4 eingestellt, der Generator erneut gestartet und die erste „echte“ Datei aufgezeichnet. Die Zufallsdaten sind nun von höchster Qualität, da der DC-Offset von Anfang an richtig eingestellt ist.

Plausibilitätsprüfung

Nach dem Abgleich nehmen Sie eine kleine 16-MB-Datei auf und analysieren Sie sie mit einem Editor wie WinHex, um ein Histogramm zu erstellen, das das Auftreten aller 256 möglichen 8-Bit-Muster darstellt. **Bild 4** zeigt ein recht stacheliges, ungleichförmiges Histogramm für Daten, die für ernsthafte Kryptographie längst nicht zufällig genug sind, obwohl ihr 0/1-Bit-Verhältnis stimmt. Nur durch die sorgfältige Abstimmung erhält man dagegen Histogramme wie in **Bild 5**.

Hüte dich vor EMI!

Hochpräzise Instrumente wie dieser Generator sind ohne richtige Gehäuse viel zu empfindlich gegenüber EMI und HF-Strahlung. Es wird daher dringend empfohlen, das RNG in einem Metallgehäuse (Aluminium) mit ordnungsgemäß angeschlossener Schaltungsmasse zu montieren.

NIST-Test

Normen zur Bewertung der Qualität eines zufälligen Bitstroms wurden von Normungsinstituten wie dem US-amerikanischen *National Institute Standards and Technology* (NIST) festgelegt. Das NIST hat 15 Tests [1] für Zufälligkeit entwickelt, die auf drei einfachen Regeln basieren:

1. Ein wirklich zufälliger Bitstrom hat eine gleiche Anzahl von Einsen und Nullen.
2. In einem wirklich zufälligen Bitstrom treten die unterschiedlichen Bitmuster mit der gleichen Frequenz auf.
3. Kein Teil des Bitstroms ist mathematisch mit irgendeinem anderen Teil korreliert.

Wenn unser Gerät so sorgfältig abgestimmt ist, dass es Histogramme wie in Bild 5 erreicht, absolviert es die 15 NIST-Tests mit Leichtigkeit. Das erste Kriterium wird durch sorgfältige Einstellung der Gleichstrom-Vorspannung mit Trimmpoti P5 erfüllt, das zweite durch Einstellen des Frequenzgangs des Korrekturfilters

mit seinen Kondensator- und Widerstandsbänken (DIP-Schalter S5, S6 und S7), den Trimmkondensatoren C39, C47 und dem Trimmpoti P6. Da die Quelle des Bitstroms ja ein zufälliges Rauschen ist, ergibt sich das dritte Kriterium automatisch. Bevor wir uns mit den NIST-Tests auseinandersetzen, können Sie selbst einen einfachen Test mit einem Komprimierungsprogramm wie 7-Zip durchführen. Die Algorithmen solcher Programme versuchen, die Redundanz oder Korrelation der Daten zu reduzieren. Da echte Zufallsdaten vollständig unkorreliert sind, können sie auch nicht komprimiert werden. Die Größe der komprimierten Datei ist daher größer als die ursprüngliche Datei (die Erhöhung beruht auf den Dateiformaten). Die Windows-Anwendung unter [2] führt NIST-Tests durch und stellt Pseudozufallszahlengeneratoren (PRNGs) für gute pseudozufällige Bitströme zum Vergleichen sowie Test-Bitströme zur Verfügung.
Die NIST-Testapplikation benötigt eine Textdatei als Eingabe, so dass die Binärdateien zuerst konvertiert werden müssen. Die Anwendung ist sehr heikel, da nur die ASCII-Zeichen „0“ (ASCII-Code 0x30), „1“ (ASCII-Code 0x31) und Leerzeichen (ASCII-Code 0x20) erlaubt sind. Die Binärzahlen müssen durch je ein Leerzeichen getrennt sein, führende oder nachgestellte Leerzeichen sind nicht zulässig. Ein Fehler ist schnell passiert und schon wird die Datei zurückgewiesen.
Wenn der Bitstrom in einem bestimmten NIST-Test fehlschlägt, hat das immer mit den drei oben genannten, grundlegenden Kriterien zu tun. Etwas zusätzlicher Abgleich des Generators sollte das Problem beheben. Wenn zum Beispiel ein Bitstrom den Monobit-Test (ein einfacher 0/1-Verhältnis-Test) erfolgreich durchläuft, aber an dem Runs-Test, der die Anzahl der Sequenzen aller Einsen oder aller Nullen zählt, mit einem überhöhten Wert scheitert, dann dürfte die gemittelte analoge Frequenz ein bisschen zu hoch sein. Dem Problem ist leicht abzuhelfen: Durch Erhöhung der mit S5 gewählten Kapazität kann die untere Grenzfrequenz leicht abgesenkt werden.
Mit einer kleinen Optimierung der Trimmereinstellungen ist es nicht schwierig, mit dem RNG erfolgreich alle 15 NIST-Tests für viel längere Bit-Streams zu absolvieren.

Fazit

Der hier beschriebene Random Number Generator (RNG) mit seinen billigen, leicht verfügbaren Bauteilen ist gut genug, um den offiziellen Standards internationaler Normungsorganisationen wie dem NIST gerecht zu werden. Dieses Beispiel zeigt, dass DIY noch lange nicht tot, sondern im wahren Geist von Elektor lebendig ist wie eh und je! Der RNG ist mehr als ein experimentelles Spielzeug, es ist ein echtes Präzisionsinstrument. Andererseits ist zu bedenken, dass Verschlüsselungsschemata dieses oder noch besserer Generatoren theoretisch mit einem Quantencomputer durchbrochen werden können. Dafür übernehmen weder Elektor noch der Autor eine Verantwortung. ◀

(150116)

Weblinks

[1] http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

[2] www.elektormagazine.com/labs/random-number-generator-150116

[3] www.elektormagazine.de/150116

Der Autor

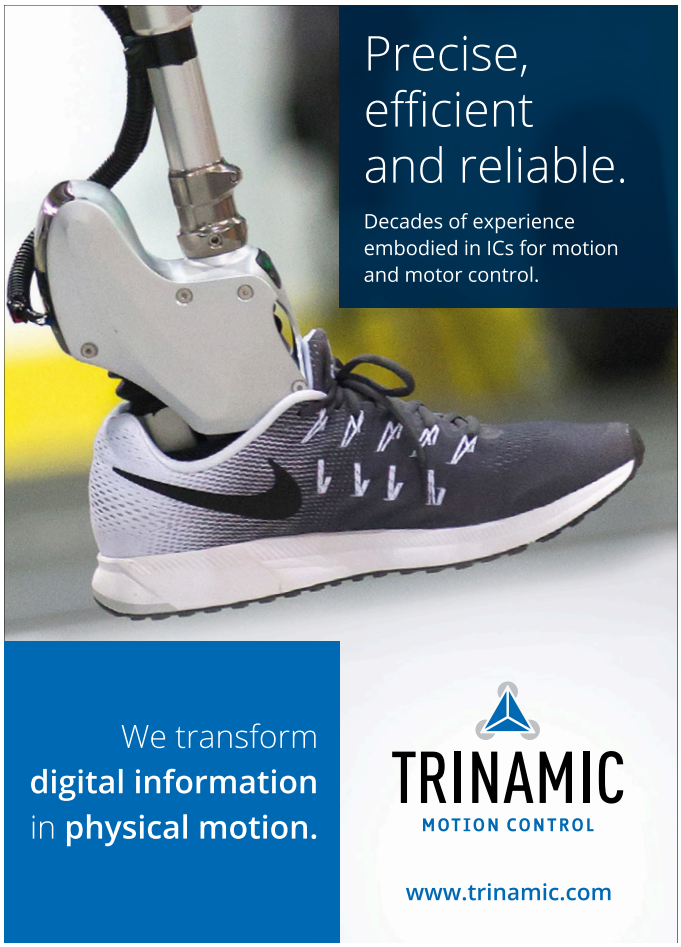
Der Autor Luka Matić absolvierte sein Studium an der FER Zagreb (Kroatien), wo er einen Master-Abschluss in Automatisierungstechnik erwarb. Beruflich beschäftigte er sich mit der Automatisierung von Industrieanlagen an Land wie auf dem Meer sowie mit ferngesteuerten Tauchrobotern (ROSVs). Als freier Ingenieur hat er viel Erfahrung mit allen möglichen elektronischen Geräten gesammelt. Luka ist nicht nur Autor bei Elektor und Elektor /Labs, sondern auch Co-Autor eines wissenschaftlichen Artikels über fortgeschrittene Reibungskompensation für elektrisch angetriebene Positioniersysteme. Man kann ihn erreichen unter luka.matic@gmail.com.

EINKAUFSZETTEL

- **150116-1** Platine
- **150116-2** Platine
- **150116-41** Programmierter Mikrocontroller
- **150116-42** Programmierter Mikrocontroller

(siehe [3])

Anzeige



Projekt 2.0

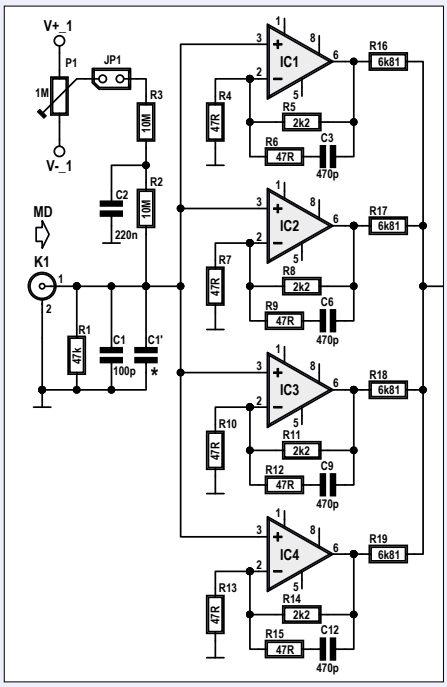
Korrekturen, Updates und Leserbriefe

SUPRA 2.0

Elektor 6/2016, S. 56 (150616)

Wenn Sie eine optimale Noise-Performance für MM-Phonotonabnehmer erreichen möchten, empfehle ich dringend nur einen LT1028 Opamp in der Eingangsstufe zu verwenden. MM-Tonabnehmer haben eine frequenzabhängige Eingangsimpedanz von ein paar hundert bis einigen kΩ. Mit einem Eingangsrauschstrom von 1 pA/Wurzel Hz wird das erzeugte Rauschen größer als das Spannungsrauschen. Mit vier LT1028 Opamps und einer Tonabnehmerimpedanz von 1 kΩ wäre das stromgenerierte Rauschen 2 nV/Wurzel Hz. Für eine Ein-Opamp-Version sollte R16 auf 1,7 kΩ reduziert werden. MC-Tonabnehmer sind niederimpedant; da macht die Vier-Opamp-Konfiguration Sinn.

Andreas von Ow



Hallo Herr von Ow,
wir stimmen Ihnen voll zu, aber hier ging es darum, einen Vorverstärker zu entwerfen, der sowohl für MC und MM gute Ergebnisse liefert. Wir haben keine alternativen Opamps gefunden, die sowohl bezüglich des Spannungs- als auch des Stromrauschens für MM und MC einen besseren Kompromiss darstellen würden.

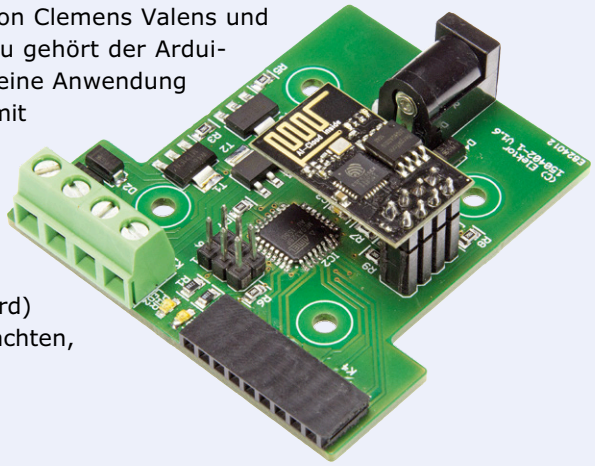
Harry Baggen und Ton Giesberts

Die Rückkehr des WLAN-Controller-Boards

Elektor 11/2016, S.22 (150402)

Meine Frage betrifft das „WLAN-Controller-Board“ von Clemens Valens und Roy Aarts, welches ich übrigens sehr gut finde. Dazu gehört der Arduino-Sketch „wifi_controller_v1.3.ino“, welchen ich an meine Anwendung anpassen möchte. Nachdem ich noch keinerlei Erfahrung mit einer In-Circuit-Programmierung von Arduino-Sketches habe, bitte ich Sie um Rat. Ich beabsichtige mir das preiswerte Modul „Arduino ISP“ (ein Arduino-Produkt) zu besorgen, welches ungefähr ab der Arduino-Version 1.5 einsatzfähig ist.
Man steckt dieses einfach auf die ISP-Pins des Ziel-Boards (hier K6) und kann dann (nach der passenden Einstellung über tools/board) wie üblich den Sketch über USB herunterladen. Was muss man beachten, zum Beispiel wegen der 3,3-V-Betriebsspannung des Boards?
Oder gibt es vielleicht eine bessere Methode?

Herbert Nitsche



Hallo Herr Nitsche,
das WLAN-Controller-Board wird mit 3,3 V gespeist, der „Arduino ISP“ ist für 5-V-Systeme ausgelegt. Es ist möglich, ihn trotzdem zu verwenden, wenn Sie die Arduino-ISP-Stromversorgung nicht mit dem ISP-Stecker K6 auf dem WLAN-Controller-Board verbinden (es befindet sich hierfür eine Lötbrücke auf dem Arduino ISP). Dann müssen Sie das Board über seine normale Spannungsversorgung speisen. Es ist vielleicht eine gute Idee, außerdem kleine 100-Ω-Widerstände in Serie mit den MOSI-, MISO-, SCK- und RESET-Leitungen zu legen, wahrscheinlich ist das aber kein Muss.

Clemens Valens

Digitale Filter einfach erklärt - FIR-Filter

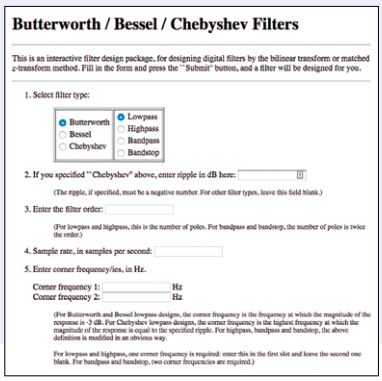
Elektor 12/2016, S. 26 (150635)

Euer Grundlagenartikel über FIR-Filter hat mir sehr gefallen. Toller Artikel von Robert Lacoste! Ein Tipp: Der leider recht jung verstorbene Tony Fisher hat ein geniales Onlinetool zum Entwurf von digitalen Filtern programmiert, zum Glück hält die Uni York die Webseite noch am Leben:

https://www-users.cs.york.ac.uk/~fisher/mkfilter/

Als Ergebnis erhält man C-Quellcode und man kann damit nicht nur FIR-, sondern auch IIR-Filter beliebiger Ordnung realisieren.

Matthias Weingart



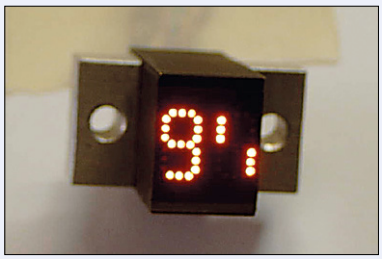
Willkommen bei Share - Nixie ohne Nixie (3)

Elektor 12/2016, S. 68 (160223)

Diese Idee ist nicht neu. Displays dieser Art gab es schon vor über 40 Jahren als Anzeige in hellem Umgebungslicht, z.B. in Cockpits von Flugzeugen.

Als Lichtquelle dienten spezielle Glühlämpchen. Das Foto zeigt ein solches Display. Es ist leider nicht mehr voll funktionsfähig.

Max Hiltmann

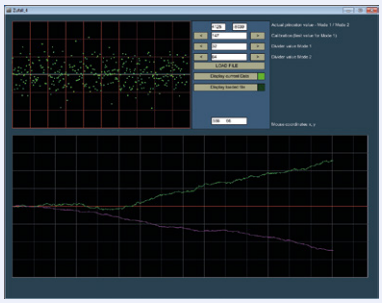


Analoge Zufallszahlen

Elektor 12/2016, S. 53 (150831)

Um der Esoterik (je genauer man misst, desto weniger positive Ergebnisse sind zu beobachten!) mit dieser schon 25 Jahre alten Messmethode zum Erfolg zu verhelfen, muss man auf Folgendes achten:

1. Alle Verstärker, Komparatoren und insbesondere die rauschende Zenerdiode mit Spannung aus dem 50-Hz-Netz versorgen, damit auch sichergestellt ist, dass sich im Rauschen ein winziger 100-Hz-Brumm (Oberwellen sind von Vorteil) versteckt.
2. Das Rausch-Brumm-Gemisch mit einem ganzzahligen Vielfachen der Netzfrequenz (quarzgenau oder besser atomuhrgenau) von 100 Hz abtasten.



Damit hat man sich einen schönen synchronen Gleichrichter (Lock-In-Amplifier) gebastelt. Wird immer auf die Mitte der (unsichtbaren) positiven Brummhalbwelle abgetastet, dann führt dies zu einem winzigen positiven Offset. Fürs Abtasten der negativen Brumm-Halbwelle gilt das gleiche.

Beobachtet man die Phase/Frequenz unseres 50-Hz-Netzes über 24 Stunden, stellt man fest, dass diese Werte nicht stabil, sondern abhängig von der Netzlast sind. Tagsüber geht die Netzuhr nach und nachts holt sie wieder auf. Treten besagte „Ereignisse“ auf, z.B. ein Fußballfinale, dann müssen die Stromversorger ihre Generatoren länger unter erhöhter Last laufen lassen und können die verspätete Phase erst nach dem Schlusspfiff aufholen.

Dieser Vorgang ist als winziger Sprung am Ausgang der esoterischen Messaparatur zu beobachten.

Wer mehr über Esoterik lernen will, sollte sich die GWUP-Homepage anschauen.

Heinz Lenk

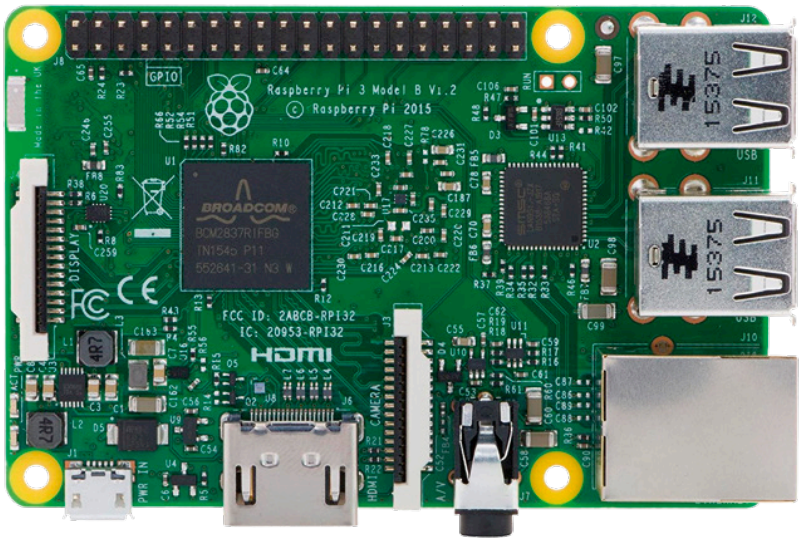
Sehr geehrter Herr Lenk,

vielen Dank für Ihren humorvollen Leserbrief, über dessen ironischen Unterton wir sehr schmunzeln mussten. Sie haben natürlich Recht, dass man sich bei der Suche nach einem „globalen Bewusstsein“ sehr schnell auf das Glatteis der Esoterik begeben kann. Wir dürfen Ihnen daher versichern, dass auch wir mit Esoterik nichts am Hut haben und unseren Lesern mit dieser Schaltung lediglich ein vielseitig verwendbares Instrument vorstellen wollten. Rauschsignale werden, wie Sie sicher wissen, in der Elektronik auf vielen Gebieten eingesetzt. Außerdem kann die veröffentlichte Software im Zusammenhang mit dem Arduino auch zur Aufzeichnung anderer Daten (wie zum Beispiel Temperaturen) verwendet werden. Mit dem Hinweis auf das Princeton-Projekt (GCP), das zweifellos existiert, behaupten wir nicht automatisch, dass wir mit den Ansichten seiner Betreiber übereinstimmen, sondern bieten dem Leser lediglich eine Möglichkeit, sich seine eigene Meinung dazu zu bilden. Elektor hat sich schon immer augenzwinkernd mit grenzwissenschaftlicher Elektronik beschäftigt; es gab früher sogar einmal ein Sonderheft dazu, weil der Verlag der Meinung ist, dass auch Hobbyelektroniker Sinn für Humor haben und nicht alles mit verbissenem Ernst betrachten.

Kurt Diedrich

Android auf dem Raspi (1)

Messen, Steuern, Regeln: GPIO-Pins



Google bringt eine neue Android-Version heraus – speziell für Einplatinen-Rechner und das Messen, Steuern und Regeln. Wir haben Android auf dem Raspberry Pi ausprobiert und zeigen Schritt für Schritt, wie man zu einem ersten „Hallo-Welt"-Projekt kommt.

Von **Tam Hanna** (Slowakei)

Microsofts rapides Voranschreiten im Bereich des Raspberry Pi hat bei Google sicher Bauchschmerzen ausgelöst: Nach dem Desktop, dem Tablet und dem Handy dürfte das Internet of Things zur nächsten Kampfarena werden.
Da die Redmonder bei der Plattformunabhängigkeit weit gekommen sind - Stichwort Konvergenz - muss Google schon aus dem Grund im IoT mitspielen, weil Microsoft sonst von dieser Richtung aus ein weiteres Mal den Handymarkt angreifen könnte.
Das vor rund einem Jahr angekündigte „Brillo" hatte wenig Erfolg. Beim zweiten Versuch geht man aufs Ganze, und benennt sein neues Produkt nur noch „Android Things".

Auf Qualität achten!

Achten Sie darauf, bei der Bestellung von Karten aus China sorgfältigste Qualitätsprüfungen durchzuführen. Viele Hersteller verkaufen chargenweise minderwertiges Material, das dem Controller gegenüber zwar den vollen Speicherausbau meldet, im praktischen Einsatz dann aber Daten verliert.

Einfacher Zugriff auf Hardware

Beginnen wir mit dem Offensichtlichen. Trotz allem Brimborium - Analogien zu Microsofts „Windows 10 IOT Core" sind rein zufällig - ist Android Things im Großen und Ganzen eine normale Version von Android, die allerdings einige Erweiterungen für die Interaktion mit Hardware enthält. In **Bild 1** sieht man die Struktur des Betriebssystems.
Die *Things Support Library* besteht dabei aus zwei Modulen. Erstens gibt es die *Peripheral I/O API*, die zum Zeitpunkt der Drucklegung einen Zugriff auf die folgenden vier Peripheriearten ermöglicht (ob OneWire nachgerüstet wird, ist im Moment unklar):

- GPIO mit PWM
- I^2C
- SPI
- UART

Der zweite Teil ist die *User Driver API*. Es handelt sich dabei um eine Programmierschnittstelle, über die Entwickler die von ihren hauseigenen Sensoren angelieferten Informationen dem Rest des Betriebssystems zur Verfügung stellen können. Weitere Informationen zu den in der Things Support Library zusammengefassten Modulen finden sich unter [1].
Wir wollen nun erste Versuche mit praktischer Hardware wagen.

Fertige Images

Zum Zeitpunkt der Drucklegung bietet Google - man hat wohl aus den Fehlschlägen mit Brillo gelernt - Images für die Plattformen Intel Edison, NXP Pico und Raspberry Pi 3 an.
Wir wollen in diesem Artikel auf den mit Abstand am weitesten verbreiteten Raspberry Pi 3 setzen. Laden Sie im ersten Schritt das unter [2] bereitstehende Image herunter, und brennen Sie es wie gewohnt auf eine Speicherkarte (siehe hierzu den **Kasten** „Auf Qualität achten!"). Der Autor nutzt in den folgenden Schritten eine 8 GB große Karte. Die Verwendung größerer Karten sollte kein Problem sein, während das Image ob seiner Größe von 4,6 GB nicht mit kleineren Karten zusammenarbeitet.

Stecken Sie im nächsten Schritt den Bildschirm und ein Netzwerkkabel zu Ihrem Router am Raspi an, und starten Sie ihn durch das Anschließen der Stromversorgung (Tastatur und Maus sind nicht erforderlich, weil Android Things von Haus aus nur einen Desktop ohne relevante Interaktionsmöglichkeiten anzeigt). Auf das Debugging über WLAN wollen wir nicht weiter eingehen, weil die höhere Latenz von drahtlosen Verbindungen bei der Fehlerjagd für Delays sorgen kann. Wer es trotzdem probieren möchte, muss den Raspi nach den unter [3] bereitstehenden Anweisungen mit dem Funknetz verbinden.
Der erste Start von Android Things dauert auch auf einem Raspberry Pi 3 die eine oder andere Minute. Ganz zu Beginn erscheint eine Meldung über fehlendes Ethernet - diese können sie bedenkenlos ignorieren, wenn das Netzwerkkabel schon angesteckt ist.
Nach getaner Arbeit präsentiert das Betriebssystem den in **Bild 2** gezeigten Startbildschirm. Die IP-Adresse – hier 10.42.0.44 – benötigen wir gleich.

Entwicklungsumgebung

Schon aus Platzgründen müssen wir an dieser Stelle davon ausgehen, dass Sie über eine funktionierende Installation von Android Studio auf Ihrem Entwicklungs-PC verfügen. Ist dies nicht der Fall, finden Sie unter [4] Rat und Hilfe. Der Autor nutzt in den folgenden Schritten eine AMD-Achtkernworkstation mit Ubuntu 14.04; unter Windows und Mac OS funktioniert es im Großen und Ganzen analog.

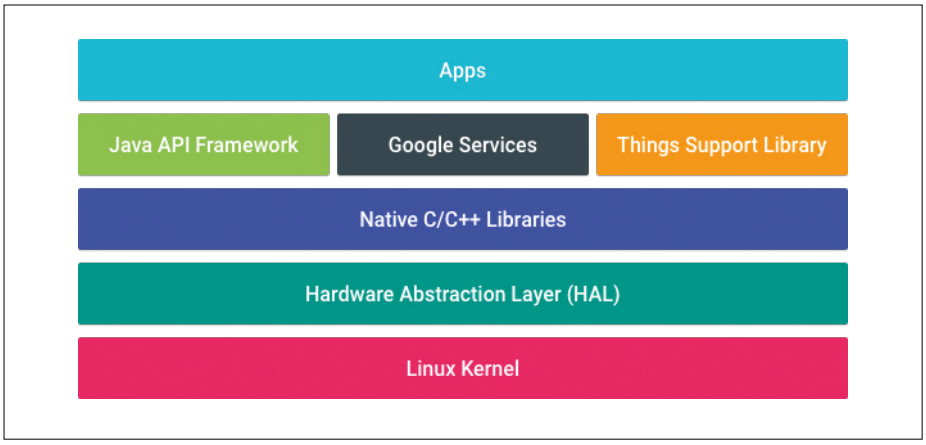


Bild 1. Android Things ist eine Variante von Android (Bildquelle: Google).

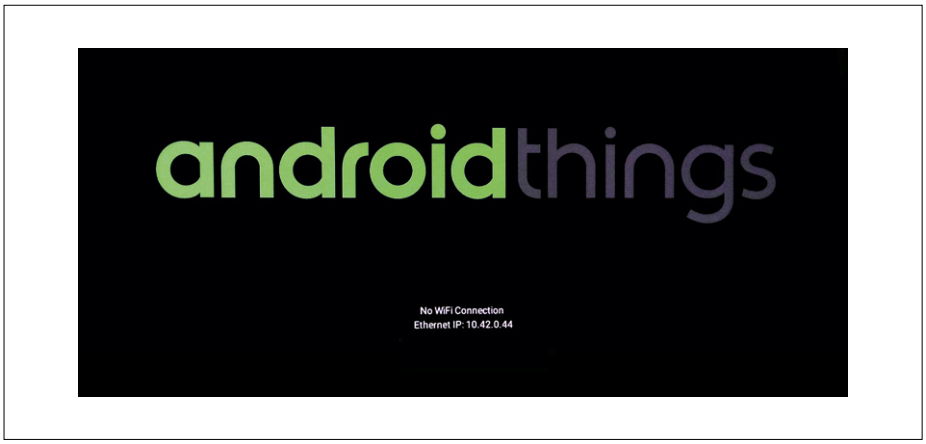


Bild 2. Unser Thing ist einsatzbereit - notieren Sie sich die IP-Adresse.

Wechseln Sie im nächsten Schritt auf dem PC in das Stammverzeichnis der *Android Debug Bridge* (ADB). Dann verbinden Sie den Entwicklungsrechner durch Eingabe der fett abgedruckten Konsolenkommandos folgendermaßen mit dem Raspi:

```
tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$ ./adb connect
   10.42.0.44

connected to 10.42.0.44:5555

tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$ ./adb devices

List of devices attached
10.42.0.44:5555 device

tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$
```

Die Eingabe von `adb devices` ist dabei nicht unbedingt erforderlich - wir zeigen Ihnen das Kommando deshalb, weil es Ihnen das Ermitteln aller mit der Android Debug Bridge verbundenen Geräte ermöglicht.
Bedenken Sie zudem, dass die Verbindung zwischen der ADB und dem Raspi verloren gehen kann, zum Beispiel wenn Ihr PC in den Schlaf fällt.

Ob des noch frühen Entwicklungsstands von Android Things gibt es derzeit keine Vorlage, die im Projektgenerator von

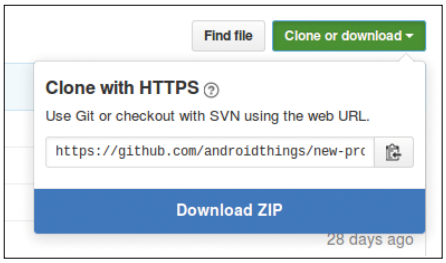


Bild 3. GitHub ist und bleibt eine Pest.

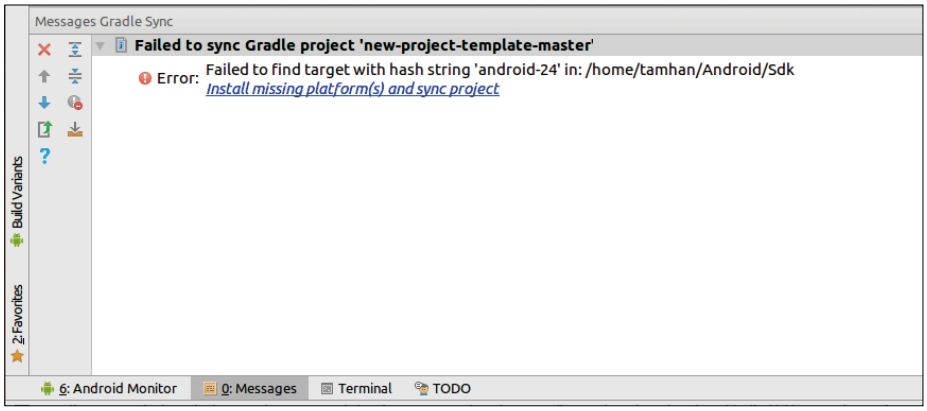


Bild 4. Hier fehlt ein kompatibles SDK.

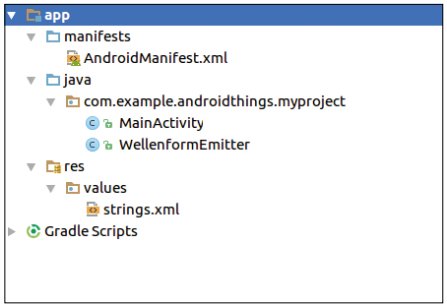


Bild 5. Diese Applikation kommt ohne XML-Markup aus.

Bild 7. Wer den GUI-Thread blockiert, wird bestraft (Bildquelle: Google).

Android Studio verfügbar ist. Sie müssen stattdessen das unter [5] bereitstehende GitHub-Repository aufrufen und dort auf den in **Bild 3** gezeigten Button klicken, um ein Beispiel-Projektskelett herunterzuladen.

Falls Sie in Android Studio derzeit ein Projekt geöffnet haben, so schließen Sie dieses durch Klick auf *File -> Close Project*. Die IDE blendet daraufhin den Willkommen-Dialog ein. An dieser Stelle müssen Sie das im vorigen Schritt heruntergeladene Archiv an einer bequem zugänglichen Stelle im Dateisystem extrahieren. Klicken Sie sodann auf *Open existing Android Studio Project* und wechseln Sie in das Verzeichnis, das dieses Projekt enthält. Nach dem Anklicken von OK beginnt die IDE mit der Bereitstellung des Projekts. Im Rahmen der Synchronisation des Gradle-Projekts versucht die IDE, fehlende Komponenten automatisch herunterzuladen. Im Fall von Problemen sehen Sie eine an **Bild 4** angelehnte Fehlermeldung - das Anklicken des Hyperlinks führt normalerweise zur Problemlösung.
Wundern Sie sich nicht, wenn sich Android Studio mehrfach über veraltete Versionen beklagt. Android Things setzt die absolut aktuellste Version mehrerer Komponenten des Betriebssystems voraus, die in mehreren Schritten installiert werden müssen.

Nach dem erfolgreichen Start der IDE sollten Sie noch einmal auf *Build -> Make Project* klicken, um eine komplette Kompilation anzustoßen. Dies ist insofern wichtig, weil für die erste Kompilierung eine Internetverbindung vorausgesetzt ist - danach haben sie eine realistische Chance, mit ihrem Projektskelett auch offline arbeiten zu können.

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
```

Bild 6. Die Tabelle zeigt die Pins des Raspberry Pi.

Alles anders oder was?

Der Code der als Startpunkt dienenden `MainActivity` unterscheidet sich insofern von normalen Android-Applikationen, als Google hier zusätzliche Logik-Aufrufe eingepflanzt hat. Diese sind deshalb erforderlich, weil Android Things auch ohne Bildschirm arbeiten kann - in diesem Fall bekommen Sie dank der Log-Calls zumindest Informationen in die Debuggerkonsole.
Beachten Sie auch, dass das *res*-Verzeichnis - wie in **Bild 5** gezeigt - leer ist: Hinter dieser Activity steht kein XML-Markup. Das wirkt sich im Code wie in **Listing 1** gezeigt aus - der normalerweise vorhandene Aufruf zum Laden von Ressourcen aus XML fehlt hier.

Interessant ist auch der Aufbau der Manifestdatei, die sich – gekürzt – in **Listing 2** präsentiert. An dieser Stelle sind zwei Änderungen interessant. Erstens sorgt der Library-Teil dafür, dass die weiter oben genannten Bibliotheken ins Projekt eingebunden werden. Zweitens bekommt die MainActivity einen

zusätzlichen *Intent-Filter* eingeschrieben, der sie als Einsprungpunkt für Android-Things-Geräte kennzeichnet.
Dies ist insofern von Relevanz, als Android Things ohne einen Programmstart auskommen muss. Der Raspi wird im Grunde genommen zu einem „One Trick Pony“, das seine Hostapplikation abarbeitet.

Ein- und Ausgabe

Für eine erste Demo wollen wir uns der GPIO-Engine zuwenden. Ein alter Spruch besagt, dass die Echtzeitfähigkeit eines Betriebssystems mit seiner Komplexität linear oder sogar logarithmisch abnimmt: Auf Java basierende Systeme sind hier besonders unangenehm, weil der *Garbage Collector* von Zeit zu Zeit für Ärger sorgt. An dieser Stelle sei schon im Voraus angemerkt, dass der Raspberry Pi eine 3,3-V-Plattform ist: Das Anschließen von 5-V-Subsystemen führt mit Sicherheit zu einem Desaster!

Damit können wir uns dem Applikationscode in der MainActivity (**Listing 3**) zuwenden. Zuerst deklarieren wir zwei weitere Membervariablen. Die Peripheral-API wird in Form eines System-Services realisiert. Ein Service ist ein meist betriebssystemweit verfügbarer Dienst: Applikationen, die ihn nutzen wollen, beschaffen sich einen Verweis und interagieren sodann mit diesem. Die `Gpio`-Klasse ist für das eigentliche Ein-und Ausschalten der Pins zuständig.
In der Funktion `onCreate` (Listing 3) erfolgt die Beschaffung des Services.
Auch wenn sich Google bei den GPIO-Funktionen nicht zu 100 % am Arduino orientiert: Eigentlich ist alles klar, was hier passiert. Die BCM-Strings lassen sich anhand der Tabelle in **Bild 6** den einzelnen Pins zuordnen. Diese Tabelle erhält man, wenn man auf einem Raspberry Pi (mit dem Standard-Betriebssystem Raspbian, nicht Android) das Kommando `GPIO Readall` eingibt.

Aber Hossa!

Zur Feststellung der Zuverlässigkeit bzw. Echtzeittauglichkeit eines Betriebssystems bietet es sich an, so schnell wie möglich eine periodisch verändernde Ausgabe am GPIO-Pin zu realisieren. Wenn man sich das Spektrum auf einem Modulationsdomänenanalysator anschaut, kann man einige Schlüsse ziehen. Rein theoretisch spricht nichts dagegen, das periodische Hoch- und Runterschalten des Pins direkt in der Funktion `OnCreated` zu implementieren, die beim Programmstart aufgerufen wird. Wer das probiert, wird aber mit der in **Bild 7** gezeigten Fehlermeldung konfrontiert.

Der Grund für dieses zugegebenermaßen etwas seltsame Verhalten liegt in einer Besonderheit des Android-Betriebssystems: Die Benutzerschnittstelle der Programme wird in einem dedizierten, als GUI-Thread bezeichneten Thread ausgeführt. Wer ihn blockiert, wird vom Betriebssystem gnadenlos mit dem „Abschuss“ seiner Applikation bestraft. Eine Endlosschleife wäre ein klassischer Blocker, der mit Sicherheit keine Gnade findet.
Zur Umgehung dieses Problems können wir unsere Routine in einen weiteren Thread auslagern.
Der einfachste Weg zur Erzeugung eines Threads ist dabei das Anlegen eines *Runnables*: Legen Sie im ersten Schritt eine neue Klasse an, die folgende Struktur eingeschrieben bekommt:

Listing 1. Gerüst der MainActivity - Startpunkt des Programms ohne User-Interface.

```
public class MainActivity extends Activity {
    private static final String TAG = MainActivity.class.
   getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, „onCreate“);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, „onDestroy“);
    }
}
```

Listing 2. Manifestdatei mit Einbindung der Bibliotheken.

```
<manifest . . .>
    <application
      - - ->
        <uses-library android:name=“com.google.android.Things“/>

        <activity android:name=“.MainActivity“>

            . . .

            <intent-filter>
                <action android:name=“android.intent.action.MAIN“/>
                <category android:name=“android.intent.category.
   IOT_LAUNCHER“/>
                <category android:name=“android.intent.category.
   DEFAULT“/>
            </intent-filter>

        </activity>
    </application>
</manifest>
```

Listing 3. Die Gpio-Klasse ist für das Ein-und Ausschalten der Pins zuständig.

```
public class MainActivity extends Activity {

    private static final String TAG = MainActivity.class.
   getSimpleName();
    PeripheralManagerService service;
    Gpio myGPIO;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, „onCreate“);

        PeripheralManagerService service = new
   PeripheralManagerService();
        try{
            myGPIO = service.openGpio(„BCM6“);
            myGPIO.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);
            myGPIO.setValue(true);
        }
        catch (Exception e){Log.d(TAG, „Fehler:“ + e.getMessage());}
```

```
public class WaveformEmitter
   implements Runnable {
    @Override
    public void run() {

    }
}
```

Runnables sind im Grunde genommen eine Art Container, der die im neuen Thread auszuführende Logik „verpackt“. Neben der hier vorgeschriebenen Methode `run` dürfen Sie natürlich diverse Member implementieren, um die für die Threadausführung notwendigen Informationen vorzuhalten.
Im Vollausbau sieht die resultierende Klasse dann wie in **Listing 4** aus.

Listing 4. Die Endlosschleife der Signalerzeugung wird in einem eigenen Thread ausgeführt – ein *Runnable* dient als Container.

```
public class WaveformEmitter implements Runnable {
    Gpio myGpio;
    public WaveformEmitter(Gpio _which){
        myGpio=_which;
    }
    @Override
    public void run() {
        try {
            while(1==1){
                myGpio.setValue(true);
                myGpio.setValue(false);
                myGpio.setValue(true);
                myGpio.setValue(false);
                myGpio.setValue(true);
                myGpio.setValue(false);
            }
        }
        catch (Exception e){}
    }
}
```

Neben einem für das Entgegennehmen der GPIO-Instanz zuständigen Konstruktor haben wir nun auch die Methode `run()` bevölkert, die für die eigentliche Wellenformausgabe verantwortlich ist. Wir erzeugen hierbei eine charakteristische Wellenform aus drei Rechtecken: dies ist insofern interessant, als die Dauer der Abarbeitung der `while`-Schleife in diesem Fall separat ausgewiesen wird. Nun stellt sich die Frage, wie wir das Runnable in `OnCreated` aktivieren. Ein klassischer Anfängerfehler wäre das direkte Aufrufen von `run()` - wer dies tut, führt den Code im Kontext der aktivierenden Methode aus (das bedeutet wieder im GUI-Thread). Der korrekte Weg zur Aktivierung eines zweiten Threads ist das Erzeugen einer neuen Thread-Klasse, mit einem als *Payload* bezeichneten Objekt als Parameter (**Listing 5**). Durch die `start()`-Methode wird der Thread angelassen.

Wir sind nun bereit, das Programm in Richtung des Raspis zu jagen. Dank der als Abstraktionsschicht fungierenden ADB genügt hierbei ein Klick auf *Run* – der RPI verhält sich wie ein per USB mit dem PC verbundenes Telefon. Da die MainActivity kein User-Interface enthält, erscheint bei angeschlossenem Monitor ein weißer Bildschirm – er informiert uns darüber, dass unsere Activity froh zu Werke geht.

Auswertung

Verbinden Sie den Raspi im nächsten Schritt mit einem Modulationsdomänenanalysator, um sich am in **Bild 8** gezeigten Schirmbild zu erfreuen (der Autor bietet unter [6] ein englischsprachiges Video an, das weitere Informationen zur Rolle und Nützlichkeit eines Modulationsdomänenanalysators bietet).
Auffällig ist neben dem gelegentlichen Jitter vor allem die Bildung zweier Höcker: Der Bereich 2,073 kHz zeigt dabei die beiden „linearen Durchläufe“ während der Übergang durch die While-Schleife zu einer etwas geringeren Frequenz von „nur“ 2,062 kHz führt.

Klar ist, dass Android hier in keiner Weise mit klassischem Unix mithalten kann; die zwischengeschaltete Java-VM fordert nun mal ihren Tribut. Mehr über das Echt-

zeitverhalten von Android kann man in diversen Papers im Internet nachlesen (zum Beispiel in [7]).

Fazit

Das soeben realisierte Programm ist ein klassisches Beispiel für Overkill im Embedded-Bereich: Ein vernünftiger Programmierer bekommt eine stabile Wellenform ganz ohne Echtzeit-Betriebssystem mit einigen Zeilen Code und einem PIC hin.

Bit Banging und Co. sind allerdings - das muss man offen zugeben - auch nicht der intendierte Anwendungsfall von Android Things. Die Plattform kann ihre Stärken immer dann ausspielen, wenn es an die Realisierung von anspruchsvollen Applikationen geht. Im nächsten Heft zeigen wir Ihnen, wie Sie Sensordaten auswerten und in einem Diagramm anzeigen können - bis dahin wünsche ich Ihnen guten Code! ⏮

(160361)

Listing 5. Start unseres Signalerzeugungs-Threads in der MainActivity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    . . .

    WaveformEmitter myEmitter=new WaveFormEmitter(myGPIO);
    new Thread( myEmitter ).start();
}
```

Weblinks

[1] https://developer.android.com/things/sdk/index.html
[2] https://developer.android.com/things/preview/download.html
[3] https://developer.android.com/things/hardware/raspberrypi.html
[4] https://developer.android.com/studio/install.html
[5] https://github.com/androidthings/new-project-template
[6] www.youtube.com/watch?v=lBLEfVUVGyU
[7] www.utdallas.edu/~cxl137330/courses/fall14/RTS/papers/4a.pdf

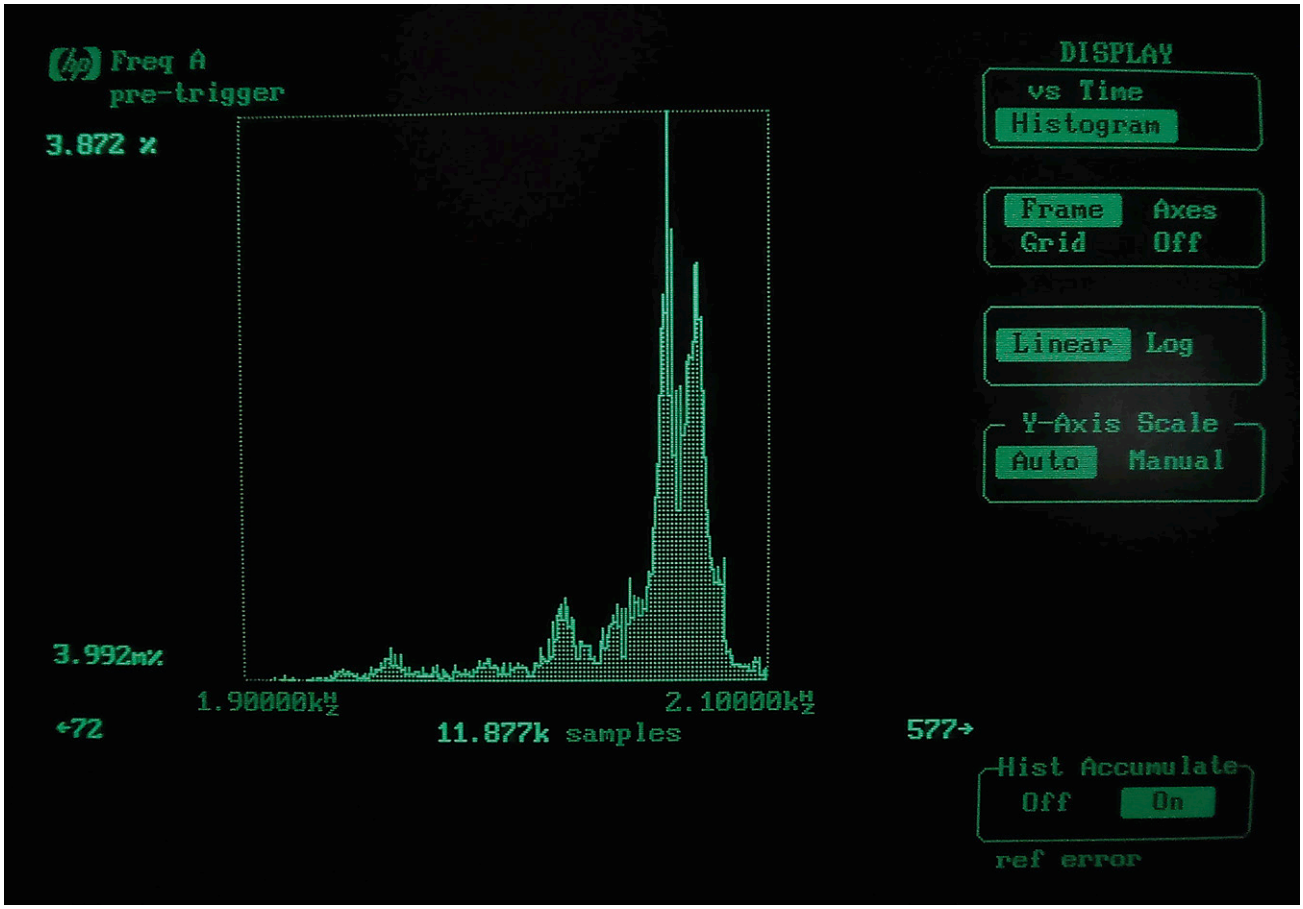


Bild 8. Arduinos schaffen hier wesentlich „schönere“ Histogramme.

Made in Belgium

Natürlich zweisprachig

In den letzten Monaten häuften sich freundliche Hinweise („Keine Lust dazu, Jan?“) von Lesern aus Belgien, dass Geräte aus ihrem Heimatland praktisch kaum in Retronik vorkommen. Nachfolgend eine Wiedergutmachung in Form der Beschreibung einiger Bausätze der einstigen Firma MBLE, die in den 1970ern Produkte für den Hobby-Bedarf im Programm hatte.

Von **Jan Buiting**

Hand aufs Herz: Wir alle hatten doch den ersten Kontakt mit Elektronik durch so etwas wie den Baukasten Philips EE (= Electronic Engineer), der im Kinderzimmer oder auf dem Küchentisch ausgebreitet wurde. Erinnern Sie sich noch an die Träume von fantastischen Apparaten, aufgebaut mit diskreten Bauteilen? Und erinnern Sie sich noch an ihren drängenden Wunsch nach Erweiterungs-Kits, die Sie prominent auf dem weihnachtlichen Wunschzettel platziert hatten? Diese Baukästen machten süchtig. Und sie waren schrecklich teuer – nicht nur als komplettes Set, sondern auch die „Ersatzteile“ (in meinem Fall: 1 Stück AC187K = 1 Woche Taschengeld).

Der Preis machte vorsichtig. Für Mutter war das Wichtigste, dass man das blinkende und piepende Zeugs nach der stolzen Demonstration wieder schön im Karton verstauen konnte.

Niederländisch, deutsch, …

Die ganze Geschichte der Elektronik-Bausätze von Philips ist ziemlich komplex und würde den Rahmen dieses Artikels sprengen. Dies hauptsächlich deshalb, weil sich der niederländische Konzern Philips in permanenten Turbulenzen befand. Dauernd wurden ganze Fabriken, Abteilungen und tausende Angestellte hin- und hergeschoben. Nachfolgend eine Kurzfassung davon, die den Hintergrund der MBLE-Kits beleuchten soll.

Elonco, die tolle Abteilung für elektroni-

sche Bauteile von Philips, begann 1955 damit, niederländische Hobby-Elektroniker mit Radio- und Audio-Bausätzen zu beglücken. Das Ingenieurbüro Amea in Den Haag erhielt den Auftrag zu Entwicklung und Design. Das Geschäft florierte und in den späten 1950ern erschienen berühmte Bausätze wie das Radio „Pionier“, der Verstärker HF302 und der Tuner FM13. Kurz darauf kamen die ersten Vielzweck- bzw. Lernbausätze wie die Typen EE8 und EE20 auf den Markt. Etwa um 1965 wurde das Team hinter den EE-Kits in Philips RGT (Radio, Grammophon, Television) eingegliedert und um ME = Mechanical Engineering erweitert [1] (**Bild 1**). Ein Teil dieses Teams arbeitete an der (tollen) Dokumentation, ein anderer an der Selektion der Bauteile (natürlich alle von Philips) und noch andere Leute beschäftigten sich mit didaktischen, grafischen und Marketing-Aspekten. Vielleicht erinnern Sie sich noch an die idealisierten Bilder „Vater und Sohn“ oder „der ideale Sohn“, die auf den Schachteln aufgedruckt waren (**Bild 2**).

Ab etwa 1970 entschieden die schwäbischen Schotten im Management von Philips Holland, dass die EE-Serie „nicht zu den Kernaktivitäten“ des Konzerns gehörte, womit das EE-Team zunächst aufs Abstellgleis geschoben wurde. Nächster Schritt war der Umzug (ohne die Leute) ins deutsche Hamburg. Ab da ging ein neues Team mit der typischen *Gründlichkeit* an die Arbeit und perfektionierte die EE-Bausätze mit großem Erfolg in einem hart umkämpften Markt. Es erschienen Kästen für Chemie, Physik und sogar Mineralogie.

Zwischenzeitlich wurden bei Philips Holland weiter Audio- und Lautsprecher-Bausätze entwickelt. Diese Aktivitäten endeten 1980. Doch die Baukästen wurden weiterhin für den niederländischen Sprachraum von Philips TSCA (Technischer Service?) lokalisiert — als Folge strotzten die Anleitungen für die Baukästen nach 1980 von drolligen Wortschöpfungen, denen man die deutsche Herkunft ansah.

Größere Perspektive

Hinter dem Gewand des Erlernens von Elektronik verbarg sich auch das Motiv von Philips, schon in jungen Jahren „die Saat für eine Markenbindung auszubringen“. Also die Gehirnwäscheabsicht, Jungs den Kauf von Philips-Bauteilen und später Geräten nahezulegen, wenn sie dereinst ihren Abschluss als Elektronik-Ingenieur – gerne an der Eindhoven Technical University – gemacht haben sollten. Die Baukästen von Philips unterschieden sich in mindestens zwei Aspekten von denen des amerikanischen Marktführers Heathkit:

1. Bei Heathkit gab es kein Zubehör und es wurden Teile von beliebigen Herstellern (manchmal umfirmiert als „Heathkit“) verwendet.
2. Die Bauteile wurden bei Philips immer wiederverwendet, denn die EE-Kits zielten auf den Bau vieler kleiner Projekte und nicht auf komplette Geräte ab.

Ich bin mir aber ziemlich sicher, dass gelegentlich ein Kit von Heathkit auf dem Schreibtisch eines Entwicklers aus dem EE-Team stand, und dass er (unwahrscheinlich: sie) ziemlich neidisch nicht nur auf die Präzision und Qualität der Anleitungen war, sondern auch auf das Resultat: Ein voll funktionierendes Gerät wie etwa einen Röhren-HiFi-Verstärker [2], ein Oszilloskop oder einen Amateurfunksender. Das alles zudem zu ziemlich günstigen Preisen. Bis in die 1980er Jahre setzte Heathkit die Maßstäbe für Bausätze und sehr sorgfältige und detaillierte Bauanleitungen sowie guten Service.

Risse im Gebälk

Trotz der happigen Preise waren die EE-Kits von Anfang an erfolgreich in ihrem Ziel, zu unterhalten, zu beeindrucken und etwas zu lehren. Doch das Brainwashing in Sachen Philips war wohl nicht ganz so erfolgreich. Elektronik als Zeitvertreib mit Karriere-Potential hatte in den 1970ern bis in die frühen 1980er zwar seine Hochphase, doch sorgte das für viel Konkurrenz, und Bauteile wie der Transistor AC187K samt Nachfolger gab es preiswert auch von anderen Herstellern in Elektronik-Läden.

Neben der erfolgreichen EE-Serie machte Philips wohl auch einen Markt für den Ansatz von Heathkit aus, also für preiswerte Geräte auf Bausatz-Basis. Philips Holland hatte hierfür zwei Asse im Ärmel:

1. Zugriff auf die enormen Bauteile-Kapazitäten des eigenen Konzerns.
2. Die Möglichkeit, eigene Audio- und professionelle Messgeräte zu kopie-

ESTD 2004

www.elektor.tv

Retronik ist eine Rubrik, die antiker Elektronik und legendären Elektor-Schaltungen ihre Referenz erweist.
Beiträge, Vorschläge und Anfragen telegrafieren Sie bitte an Jan Buiting (editor@elektor.com).

Bild 1. ME-Mechanik-Baukästen von Philips, ca. 1965.

Bild 2. Die Elektronik-Baukästen von Philips betonen die Karriere-Aussichten des schlauen und braven jungen Elektronikers.

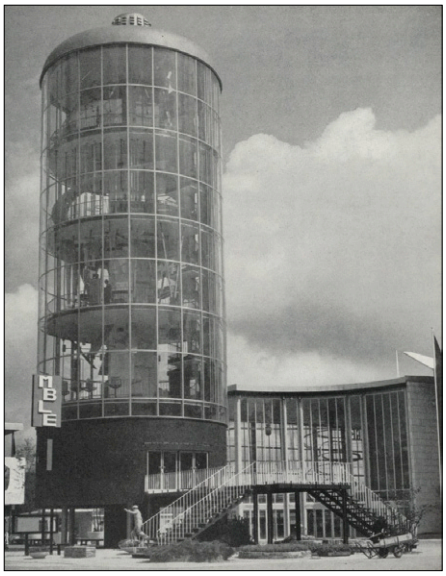


Bild 3. Der MBLE-Pavilion auf der Weltausstellung 1958 in Brüssel (im gleichen Jahr veranstaltete Philips Holland die weltweit erste Multimedia-Show namens *Poème Electronique* im eigenen, von Le Corbusier gestalteten Pavillon).

Bild 4. Der Signalgenerator BEM004. Das Gerät ist kein Fertigprodukt, sondern aus einem Bausatz von MBLE zusammengebaut.

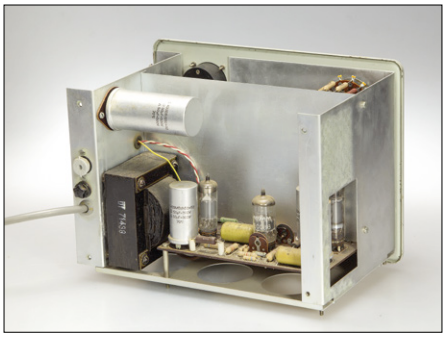


Bild 5. Innenleben des BEM004: Konventionelle Technik mit Röhren und Chassis.

ren und abzuspecken. Hierzu später ein Beispiel.

Und so wurden dann auch Bausätze entwickelt, von denen mich der Audio Mixer am meisten beeindruckte. Man erkannte die bis in die 1990er Jahre ausgelieferten Bausätze leicht an den blau/weiß bedruckten Kartons. Die Bausatz-Preise waren enorm hoch, wenn man sie mit dem vergleicht, was damals von Elektor oder anderen Zeitschriften geboten wurde, oder was die reinen Teile kosten. Die Qualität allerdings war immer erste Sahne.

Es gibt unter [3] schöne Übersichten über die Aktivitäten von Philips in diesem Segment – allerdings in niederländischer Sprache. Deutschsprachige Leser können ihre Nostalgie-Bedürfnisse unter [4] befriedigen.

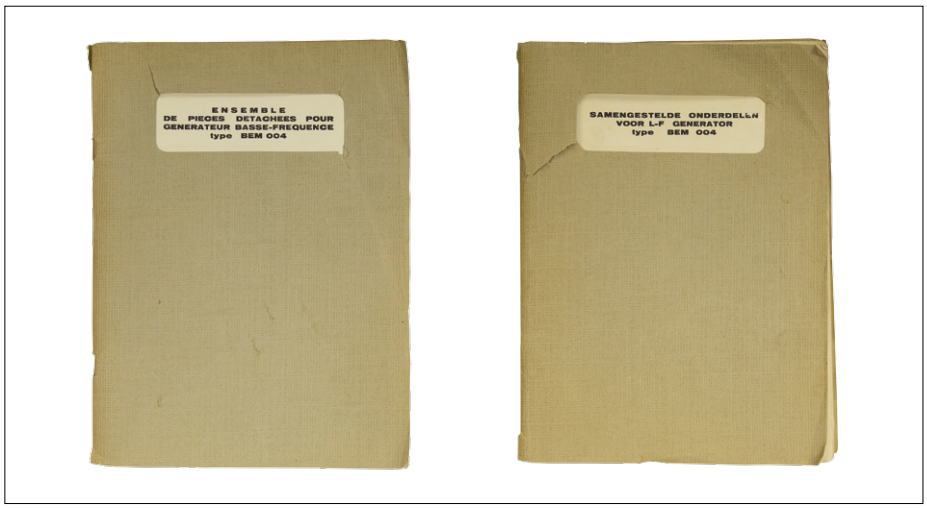


Bild 6. Zweisprachiges Ergonomie-Wunder: Die Bauanleitung zum BEM004 ist in flämischer und französischer Sprache abgedruckt – man muss das Ganze nur umdrehen.

Bild 7. Das Oszilloskop BEM005. Auch dieses Gerät ist das Resultat eines Bausatzes.

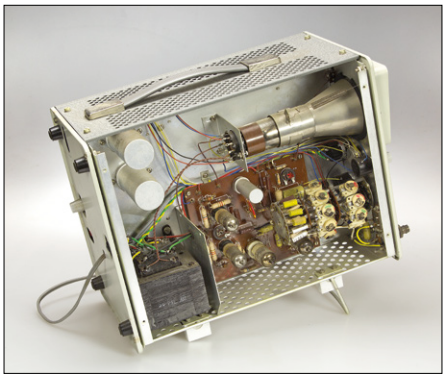


Bild 8. Innenleben des BEM005: Viel Platz und „schon" mit richtigen Platinen. Die Röhren stecken in hochqualitativen Keramik-Sockeln. Die Lötarbeiten wurden sorgfältig ausgeführt.

Belge de Lampe et de Materiel Electronique

Nach dem Verschieben der Labor-Bausätze zu RGT verschwanden sie in der Versenkung. Dann aber wurden sie wiederbelebt und als Nebenbeschäftigung an die Firma MBLE mit Hauptsitz in 80 Rue des Deux-Gares, Brüssel-7 abgegeben. MBLE wurde von der großen Philips-Subdivision Elonco (Electronische Onderdelen) geschluckt und in die Produktion von Spezialwiderständen (VDR, PTC), den berühmten senfgelben Kondensatoren sowie Elkos und sogar Röhren (Ultron) in den diversen Produktionsstätten einbezogen. MBLE war ein Mittelständler, dessen Firmengeschichte unter [5] zu finden ist. Das Bild ihres hoffentlich nicht evakuierten Pavillons auf der Weltausstellung 1958 in Brüssel (**Bild 3**) sieht aus wie die größte je gebaute Röhre des Typs PL500. Da MBLE nach der Übernahme quasi auf Bauteilen saß, wurden einige interessante Bausätze für das Heim-Labor entwickelt. Zwei Retronik-Fans aus dem flämischen Teil von Belgien waren früher Lehrer für Elektronik. Sie schickten mir einige dieser aus MBLE-Bausätzen entstandenen Geräte. Hier sind sie:

BEM004 NF-Generator

NF bzw. Niederfrequenz ist eine kleine Untertreibung für diesen mit Röhren bestückten Generator, denn seine Frequenz ließ sich in fünf Bereichen von 10 Hz bis immerhin 1 MHz einstellen. Die Schaltung enthält nichts Besonderes. Sie basiert auf üblichen Miniwatt-Röhren von Philips der Typen EF86, ECC88, EL83 und EL85 sowie Glühlämpchen im Oszillator. Im Grunde ist es eine Sparversion eines professionellen Laborgeräts aus der Serie GM56xx. Die **Bilder 4** und **5** zeigen den perfekten und sauberen Aufbau des mir anvertrauten Modells BEM004. Aus heutiger Perspektive sehen die Kunststoffknöpfe etwas billig aus, aber damals waren sie vergleichsweise prima und modern. Die Verwendung des UHF-Sockels SO-239 für das Ausgangssignal mag verwundern, denn immerhin steht AUDIO GENERATOR auf der Frontplatte.

Das Handbuch zum BEM004 (**Bild 6**) ist ein vorbildhaftes Lernmittel mit sehr detaillierten, schrittweisen Anweisungen für den Zusammenbau. Hinzu kommen große, ausklappbare Diagramme und Explosionszeichnungen des Chassis. Das Ganze ist typisch belgisch (oder gar brüsselisch), nämlich zweisprachig — man dreht die Anleitung um und hat sie dann entweder französisch oder flämisch (ein niederländischer Dialekt) vor sich. Ich kann sie in beiden Sprachen lesen und weiß daher, dass die französische Variante direkt von den MBLE-Ingenieuren geschrieben wurde, während die flämische Version ziemlich nach mehrfacher wörtlicher Übersetzung mit Mängeln im Satzbau und der Wortreihenfolge aussieht. Immerhin nicht ganz so schlimm wie eine Google-Übersetzung chinesischer Provenienz.

Die Bestückung des BEM004 mit einem Dual-Elko 50+50 µF und 400 V schreit förmlich nach modernem Ersatz (4 x 100 µF / 250 V). Die größte gemessene Verzerrung des Sinussignals betrug 0,2 % bei 100 kHz, war also innerhalb

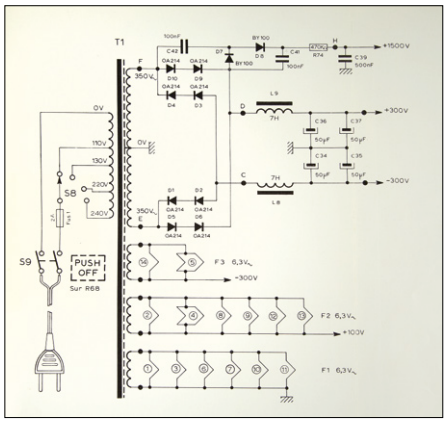


Bild 9. Mit so einem einfachen, ungeregelten Netzteil ergeben sich bei einem Oszilloskop Brummen und sonstige Störungen sowie eine generelle Instabilität des Elektronenstrahls (Reproduktion aus einem BEM003-Handbuch).

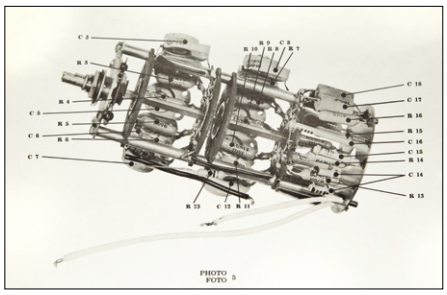


Bild 10. Zur Erleichterung des komplexen Aufbaus enthielt das Handbuch retuschierte Fotos auf extra Blättern.

Verzögerter Trigger

Seit dem Start im Jahre 2004 erfreuen sich Artikel der Rubrik *Retronik* großer Beliebtheit. Als Reaktionen gab es einige Geräte-„Spenden" und andere Beiträge, jeweils mit Hinweisen auf den originalen Artikel, der zur Reaktion veranlasst hat. Zum Schutz der Privatheit werden Namen nur dann genannt, wenn explizit die Erlaubnis zum Abdruck gegeben wurde.

Objekt 01: LCR-Brücke 1608-A von General Radio.

Dieses seltene Stück erreichte mich von einem Leser nach einem Chat mit einem unserer Übersetzer der niederländischen Elektor-Ausgabe auf einer Messe in Utrecht (NL) im Jahre 2016. Das Gerät ist eine nicht-portable Version des teureren LCR-Meters 1650-A von GenRad mit 0,1 % Genauigkeit, das in Elektor Juni 2007 besprochen wurde. Video: https://youtu.be/YfrW79pBbs4. Verglichen mit dem 1650-A ist das Modell 1608-A ein wahres Schwergewicht. Status: Braucht Restaurierung und Kalibrierung.

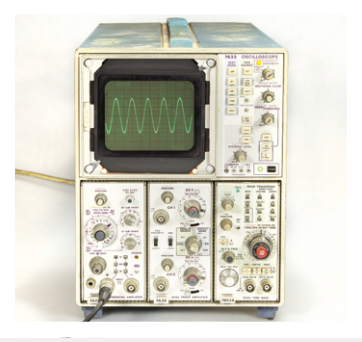
Trigger 01: Impedanz-Brücke 1650-A (1960), Elektor Juni 2007.

Objekt 02: SC/MP-Computer von Elektor.

Mir war weder die Bekanntheit noch die Bedeutung des SC/MP (Aussprache: /skamp/) klar. Vielleicht auch deswegen, weil das einzige bei Elektor übrig gebliebene Exemplar nicht modular war. Friedrich Bollow schickte mir einen „echten" SC/MP mit 3 x 4 KB RAM, serieller I/O und geladenem Elbug. Das Ganze war professionell in ein Rack-Gehäuse eingebaut. Die Boards mit CPU und RAM kosteten damals im Jahre 1978 gut 1.000 DM! Friedrich hat sogar liebevoll seine Software-Schallplatten aus der ESS-Reihe mit 45 Umdrehungen aufbewahrt, auf die Systemsoftware und andere Utilities gepresst sind. Status: Muss noch booten.

Trigger 02: SC/MP-Computer (1978), Elektor April 2005.

Objekt 03: 100-MHz-Speicher-Oszilloskop Tektronix 7633.



Unter den Herstellern von Röhren-Oszilloskopen ist Tektronix sicher der Star, wenn man die fast ein Dutzend Geräte bedenkt, die ich seit meinem Artikel über das Modell 564B erhalten habe. Das Modell 7633 konnte ich gegen ein paar Flaschen Bier, ein Buch und einige Mikrocontroller-Boards auf der Messe Elektor Live in Hanau im Jahre 2013 eintauschen. Nach der Reinigung eines Schalters und dem Ersatz von zwei Kondensatoren lief das 7633 wieder und ist nun Teil meines „klassischen" Reparatur-Arbeitsplatzes. Status: Kalibriert.

Trigger 03: Speicher-Oszilloskop Tektronix 564B (1963), Elektor Mai 2011.

Wenn Sie auch getriggert sind und hier ein Gerät auflisten lassen wollen, schreiben Sie eine E-Mail an **editor@elektor.com**, Betreff: **Trigger [xyz]**.

Bild 11. Das Oszilloskop BEM003 war das Spitzenmodell von MBLE. Leider ist es defekt und muss erst noch repariert werden.

der Spezifikation von 0,3 %. Auch die Anstiegszeit des Rechtecks ist mit 75 ns gut (im Gegensatz zur Abfallzeit).

Oszilloskop-Kits BEM003 und BEM005

Ende der 1960er Jahre war für viele Elektronik-Interessierte ein Oszilloskop ein Traum. Mit diesem magischen Instrument sollten alle Geheimnisse sichtbar werden. Leider war es nicht nur gefährlich, sondern sehr anspruchsvoll, so ein Gerät nach Schaltplänen zu bauen. Fertige Exemplare waren finanzielle Utopie. Eine zumindest theoretische Möglichkeit war, sein Geld für einen Bausatz zusammenzukratzen. 1975 gab es von MBLE folgende Oszilloskop-Bausätze auf Röhrenbasis:

- BEM003, 0...7 MHz, V: 20 mV bis 50 V/Div.; H: 0,2 µs bis 20 ms; Preis HFL 945.
- BEM005, 0...3,5 MHz; V: 50 mV bis 50 V/Div.; H: 0,5 µs bis 20 ms; Preis HFL 645.
- BEM009, 0...0,7 MHz; V: x1/x10/x100 Stufen; H: 10 Hz bis 100 kHz; Preis HFL 430.

Ein niederländischer Gulden war damals etwa 0,97 DM wert – nach heutiger Kaufkraft wären 430 HFL gut 520 €. Selbst der Preis für das Modell BEM009 war also nicht gerade niedrig, weswegen so ein selbstgebautes Oszilloskop eher selten bei einem Studenten oder „Bastler“ zu sehen war.

Ich erhielt die Modelle BEM003 und BEM005 (**Bilder 7** und **8**). Letzteres konnte ich zu neuem Leben erwecken. Nachdem ich es langsam über einen Regel-Trafo hochgefahren hatte, war es voll einsatzfähig. Doch nachdem ich seinen Stecker am nächsten Tag direkt in eine Steckdose steckte, fiel es komplett aus. Nicht einmal die Kontrolllampe wollte leuchten. Beim kurzem Check fiel mir eine völlig korrodierte Netzsicherung auf. Nachdem der Übeltäter samt Sicherungshalter gereinigt war, lief es wieder. Die vertikale Ablenkung war etwas daneben, und der zugehörige Drehschalter brauchte einige Spritzer Tuner 600 für die Reinigung der Kontakte auf zwei Ebenen. Ganz ehrlich: Der Apparat kann wirklich nicht viel. Aber vielleicht habe ich aufgrund meines Tektronix 502 zu hohe Erwartungen. Der Strahl ist nicht wirklich scharf, bewegt sich, bläht sich auf und schrumpft wieder, wenn man an den Einstellern für Intensität, Fokus oder H und V dreht. Ein klares Zeichen einer ungenügend stabilen Spannungsversorgung. Tatsächlich gibt es nicht eine einzige 85-V- oder 150-V-Stabilisierungsröhre (siehe die Netzteil-Schaltung von **Bild 9**). Die fehlende Regelung beeinträchtigt auch die Hochspannung von +1,5 kV an der Bildröhren-Anode, was zu diesen Unschönheiten führt. Das ist auch der Punkt, wo sich BEM005 und BEM003 von professionellem Equipment wie etwa der Philips-Serie GM56xx deutlich unterscheiden. Entschuldigend sollte man aber werten, dass damals Röhren und Trafos zu den teuersten Bauteilen gehörten. Wenn man also sparen musste, dann zuerst daran. Nur bei Tektronix nicht ;-). Hochmodern allerdings war, dass die Bausätze BEM003 und BEM005 mit „gedruckten Schaltungen“ aka Platinen ausgestattet waren. Damals waren das diese braunen Pertinax-Platinen von Philips. Glücklicherweise gab es Keramik-Sockel für die Röhren und gelötet hatte man ja selbst.

Auch das BEM005-Handbuch ist bilingual (französisch/flämisch), aber nicht im Umdreh-Format gedruckt wie das des BEM004. Der mechanische Aufbau und der Verdrahtungsplan ist auf gigantischen Aufklappseiten mit 65 x 45 cm zu sehen. Man konnte sie locker als Poster im Zimmer gebrauchen. Technische Fotos wie das in **Bild 10** zeigten den recht komplexen Aufbau und verhalfen nicht unwesentlich zu einem funktionierenden Oszilloskop. Der frühere Besitzer fand es auch klug, die Datenblätter aller Röhren einzuheften.

Bild 12. Beim BEM045 handelt es sich um eine Art Koffer-Elektronik-Labor von MBLE. Es war nie ein Bausatz o.ä., sondern von Philips als Hilfsmittel für den Kurs PRACTRONICS für Audio und Akustik vorgesehen.

Das größere Exemplar BEM003 (**Bild 11**) hat immer noch ein paar Macken, mit denen ich mich herumschlage. Schalter sind oxidiert, Elkos lecken und die Zeitbasis sorgt in einigen Bereichen für eine Verkürzung oder gar für ein Verschwinden des Strahls. Im Moment muss das BEM003 noch warten und dient derweil als Türstopper.

BEM045: Kein Kit sondern ein Studenten-Labor

In **Bild 12** ist mit dem BEM045 eine aufklappbare, graue Kunststoff-Box zusehen, die sich am Griff öffnen lässt. Im linken Teil befinden sich:

- ein Sinus/Rechteck-Generator mit 20...20.000 Hz und 0...1 V,
- Stromquellen mit 2 und 20 mA,
- ein von 0 bis 12 V einstellbares Netzteil und
- ein (quasi-)symmetrisches Netzteil mit 6-0-6 V.
- Im linken Teil befindet sich auch die Halterung für Batterien. Im rechten Teil steckt ein Multimeter mit:
- Amperemeter 0,1...300 mA DC,
- Voltmeter 10 mV / 30 mV / 0,1...30 V AC,
- Voltmeter 0,1...30 V DC und
- einem großen Drehspulinstrument mit Volt- und dB-Skalen.

Es dürfte klar sein, dass das BEM045 für Audio-Elektronik bzw. für Experimente im Bereich Akustik gedacht war – früher eine sehr beliebte Freizeitbeschäftigung. Ursprünglich war das portable Kit von Philips TSCA für einen Kurs mit der Bezeichnung PRACTRONICS vorgesehen, muss aber irgendwie in das Blickfeld der Marketing-Leute von MBLE geraten sein, die sein „studentisches Potential" erkannt haben. Tatsächlich ist vorne auf dem Handbuch der Schriftzug ELA EDUC PRODUCTS aufgedruckt. Unter ELA verstand man damals so etwas wie Elektrische Lautsprecher-Anlagen, im weitesten Sinn also Elektroakustik. In einem niederländischen Forum mit Beiträgen auch aus Belgien [4] gibt es Hinweise auf andere Hintergründe. Angeblich hat MBLE das Kit BEM045 speziell für die ELA-Abteilung hergestellt. Diese Logik wird gestützt von der Tatsache, dass kein BEM045 je als Baukasten auf den Markt kam oder in einem MBLE-Prospekt beworben wurde. Dieses „Labor in der Butterbrotdose" taucht aber gelegentlich noch auf niederländischen Märkten sowohl online als auch real auf.

Auch wenn die Geschichte des BEM045 nicht ganz entschlüsselt ist, bleibt es doch ein bemerkenswertes Teil, selbst wenn es nur für „Lernprojekte" nach „Lektionen" eines Buches vorgesehen sein sollte (siehe **Bild 13**).

Auf dem Gehäuse meines BEM045 stehen zwei große Zahlen. Einmal „78", geschrieben mit einem gelben Buntstift, und dann noch „39", eingebrannt mit einem Lötkolben. Beides indiziert eine Verwendung in Schulen.

Die Kiste war von auslaufenden Batterien (sechs 1,5-V-Zellen des Typs C) verschont geblieben. Der Vorbesitzer hat nämlich eine BNC-Buchse auf der linken Schmalseite eingebaut, um ein externes 12-V-Netzteil anzuschließen. Dieses Netzteil war glücklicherweise Teil des Geschenks.

Alle Teile des Mini-Labs bis auf den Generator funktionierten. Dieser brauchte eine Weile nach dem Einschalten und auch nach jeder Frequenzänderung musste sich erst seine Amplitude stabilisieren. Vermutlich ist also etwas in der Regelung defekt. Noch vermutlicher ein Elko. Merkwürdigerweise schien das Messgerät auch ohne Stromversorgung (die 12 V versorgen lediglich die linke Seite) zu funktionieren. Eine genauere Inspektion förderte den Grund zutage: Die 12 V werden von links über die zwei Gelenke nach rechts geführt. Genau diese Konstruktion ist auch die Achillesferse des BEM045.

Ist das Kit brauchbar? Begrenzt. Ich habe kein Handbuch. Das Messgerät dieses Plastik-Bombers kann Widerstände höchstens über R = U/I messen. Außerdem befürchte ich, irgendetwas durch zuviel Strom zu himmeln. Das Instrument ist zwar groß, lässt aber Skalendetails vermissen, und die Bereichswahl ist kontraintuitiv. Ein Frequenz-Sweep zur Analyse von Filtern fehlt. Positiv ist, dass das Sinussignal nach Stabilisierung recht klirrarm und die Frequenzeinstellung ausreichend genau ist. Für die damalige Zeit reicht das Gebotene gerade so für einfache Audio-Messungen.

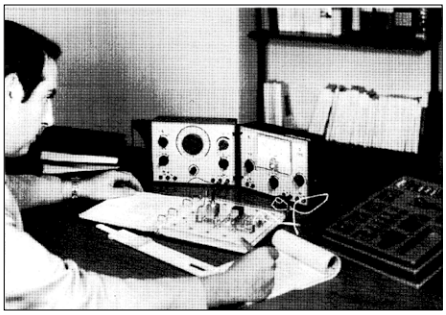


Bild 13. Der PRACtical Elektronik-Kurs von Philips, vermutlich von 1975. Mit eingebautem Signalgenerator und Multimeter war das Modell BEM045 ein Hilfsmittel zum Bau und zum Verständnis von Audio-Elektronik. Die Schaltung ist im wahrsten Sinne des Wortes auf der Anleitung aufgebaut. Bitte entschuldigen Sie die schlechte Qualität, aber es handelt sich um seltenes Original-Material.

Wozu?

Eine vor allem von Nichttechnikern mehrfach gestellte, gut gemeinte Frage war: *„Konntest Du diese Oszilloskope mit dieser Messbox reparieren?"* Natürlich nicht. Das BEM045 eignet sich nicht für Spannungen über 30 V. Auch sein Sinussignal ist nicht gut und stabil genug. Damit z.B. eine problematische Vertikalablenkung zu checken ist Sisyphus-Arbeit. Auch Einflüsse von schwankender Netzspannung lassen sich so nicht gut erfassen.

Kurz: Ich verlasse mich auf mein gutes Oszilloskop 7633 von Tektronix, einen stabilen und gut einstellbaren Regeltrafo von MBLE (!) und ein Labornetzteil von Delta, um die minimalen Pegel zu erreichen, die einen Strahl in einer Bildröhre erzeugen. Damit kann man gelegentlich sogar hartgesottene Kollegen für eine Minute beeindrucken, die sich normalerweise in Embedded-Welten tummeln. Und natürlich ein Stück Elektronik-Geschichte aus Belgien erzählen. ⏮

(160303)

Weblinks

[1] ME-Bausätze von Philips, www.elektormagazine.de/070277

[2] Röhrenverstärker AA-100 von Heathkit (1960): www.elektormagazine.de/140333

[3] www.hansotten.com/electronic-kits/ee-series/ee8-ee20-a20/

[4] Norbert Old: http://norbert.old.no/kits/ee2000/index.html

[5] MBLE: www.radiocollection.be/fr/MBLE1_fr.html

[6] MBLE045: www.philipsradios.nl/forum/index.php?mode=thread&id=23681

(Fast) alles, was Sie schon immer wissen wollten über ...

Elektromotoren

Mit jedem Elektromotor kann man etwas bewegen. Aber jeder Elektromotor weist auch spezifische Eigenschaften auf, so dass es sinnvoll ist, für die geplante Bewegung den richtigen Motortyp auszuwählen.

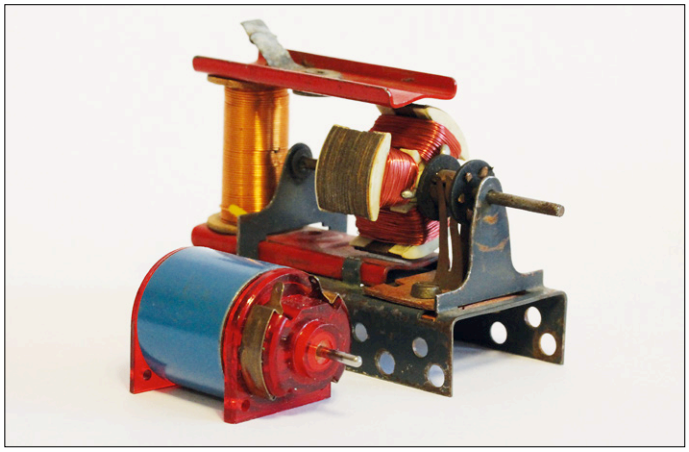


Bild 1. „My first motor“, oder eigentlich meine beiden ersten. Der blaue ist ein Gleichstrommotor, der auf der rechten Seite kann sowohl mit Gleich- als auch mit Wechselstrom betrieben werden, so dass ich ihn direkt an einen Märklin-Trafo anschließen konnte.

Von **Ties Wijffels** (Niederlande)

Q *Um mit der Versorgung zu beginnen – Gleich- oder Wechselspannung?*

A Die treibende Kraft jedes Elektromotors (**Bild 1**) ist der Magnetismus: Ein Magnetfeld treibt das andere an. Für eine permanente Bewegung muss sich eines der Magnetfelder kontinuierlich ändern. Dafür wird immer ein Elektromagnet eingesetzt. Das statische Feld kann mit einem Permanentmagneten, einem zweiten Elektromagneten oder einer kurzgeschlossenen Spule im Rotor erzeugt werden.
Ein Gleichstrommotor dreht die Pole des Rotorfeldes mit einer Art Wechselschalter, einem Kommutator, der auf dem Rotor sitzt. Es gibt aber auch einen elektronischen Weg, bei vielen Motoren sogar einen externen, etwa bei bürstenlosen (Brushless-)Motoren, wie sie in der Modellfliegerei sehr populär sind, oder bei Schrittmotoren. Bei Wechselstrommotoren sorgt die 50-Hz-Netzspannung oder eine von einem Frequenzregler abgeleitete Spannung für die Umkehrung der Polarität. Es gibt aber auch Wechselstrommotoren mit Kohlebürsten und Kommutator.

Q *Was mache ich, wenn es besonders einfach werden soll?*

A Der am häufigsten eingesetzte Elektromotor ist der Gleichstrommotor (**Bild 2**). Die kleineren Versionen besitzen meistens einen Stator mit Permanentmagneten und einen Rotor, der auf einen Weicheisenkern gewickelt ist und dem die Energie über (Kohle-)Bürsten und einen Kommutator zugeführt wird.
Sehr angenehm ist beim Gleichstrommotor, dass seine Drehzahl zur angelegten Spannung nahezu proportional ist.
Obwohl Gleichstrommotoren mit einer bestimmten Drehzahl für eine gegebene Spannung spezifiziert werden, ist diese Angabe keine „harte“ Eigenschaft. Man kann mit einer in Grenzen höheren Spannung eine höhere Drehzahl erzielen und mit einer geringeren Spannung eine niedrigere.

Q *Wie schließe ich einen solchen Motor an meiner Elektronik an?*

A Ein Gleichstrommotor wird in der Regel mit einem PWM-Signal angesteuert. Man schaltet den Motor dauernd an und aus und das Massenträgheitsmoment des Rotors stabilisiert die Drehbewegung.
Wenn die PWM-Frequenz zu hoch ist, gibt es Probleme mit der Selbstinduktion des Rotors. Deshalb sollte man entweder eine niedrige PWM-Frequenz von einigen hundert Hertz oder aber eine besonders hohe PWM-Frequenz von 20 kHz und mehr wählen.
Die Wahl des Schalttransistors ist abhängig von der gewünschten Schaltfrequenz und der erwarteten Motorleistung. Bei Kleinmotoren mit niedriger PWM-Frequenz reicht ein einfacher Leistungstransistor, für größere Motorströme sind Darlington-Arrays ideal. Für ganz schwere Motoren sollte man fast immer Leistungs-MOSFETs verwenden. Dank ihres sehr geringen Widerstands im eingeschalteten Zustand und durch den erlaubten hohen Drainstrom (100 A und mehr sind keine Ausnahme) kann die Verlustleistung (Wärme) viel geringer gehalten werden als bei einem Bipolartransistor.
Wichtig: Verwenden Sie immer eine Freilaufdiode! Das Abschalten der Rotor-Selbstinduktion verursacht immer eine erhebliche Überspannung, die über die positive Versorgungsspannungsleitung abgebaut werden muss.

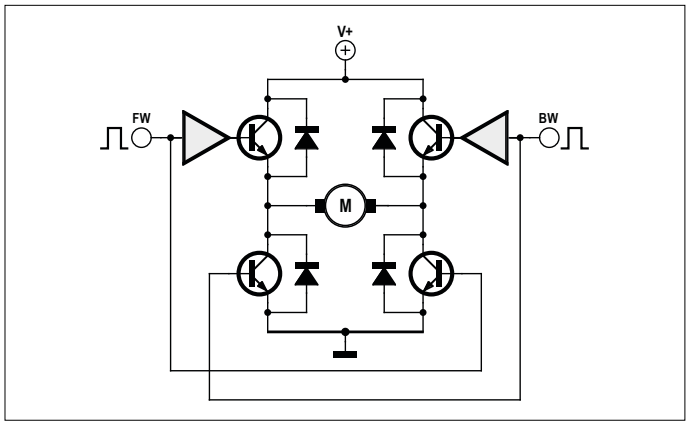


Bild 3. Ein Gleichstrommotor in einer H-Brücke kann in beide Richtungen drehen.

Bild 2. Gleichstrommotoren, ganz groß und ganz klein.

Bild 4. Bürstenloser Gleichstrommotor mit Regler.

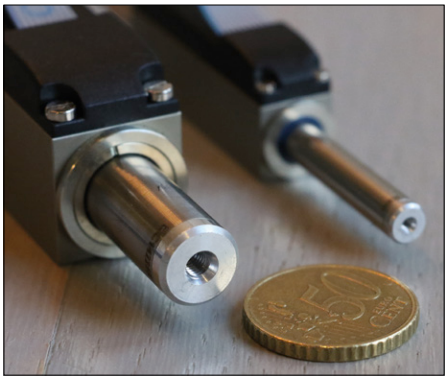


Bild 5. Encoder selbstgebaut – auch das geht!

Q *Wie kann ich die Drehrichtung umkehren?*

A Möchten Sie die Drehrichtung eines Gleichstrommotors umkehren, muss die Motorspannung umgepolt werden. Soll dies nur ab und an geschehen, reicht dafür ein simples Umpolrelais. Für kontinuierliche Polaritätsumkehr beispielsweise in einem Servosystem ist eine H-Brücke erforderlich. Dabei befinden sich vier Transistoren in einer Brückenschaltung, von denen immer zwei diagonal gegenüberliegende leiten (**Bild 3**). Auch in H-Brücken sind (hier vier) Freilaufdioden erforderlich.

Q *Und wenn ich im Hinblick auf einen niedrigeren Schlupf einen bürstenlosen Motor einsetzen möchte?*

A Dann können Sie auf einen Motortyp zurückgreifen, der viel im Modellbau eingesetzt wird und drei Anschlussdrähte besitzt (**Bild 4**). Eigentlich handelt es sich dabei um das Äquivalent eines bürstenlosen Wechselstrommotors, allerdings wird das Stator-Drehfeld extern erzeugt. Für einen solchen bürstenlosen Motor können Sie am besten einen handelsüblichen Controller verwenden. Die Ansteuerung erfolgt mit Impulsen von 1 ms (stopp) bis 2 ms (volle Leistung). Die Frequenz der Steuerimpulse kann variieren, jedoch ist man mit 50 Hz fast immer gut beraten. Zum Ändern der Drehrichtung vertauscht man zwei der drei Phasen. Wenn Sie die Drehrichtung sehr häufig umkehren wollen, empfiehlt sich ein Controller mit Nullstellung: 1,5 ms: Stopp, 1 ms rückwärts, 2 ms vorwärts.

Q *Der Motor dreht nun, aber wie steht es mit einer genauen Positionierung?*

A Die Positionierung kann auf unterschiedliche Weise erfolgen. Bei einem normalen (DC-)Motor muss die Positionsrückmeldung über einen Encoder oder wie bei den üblichen Modellbauservos über ein Potentiometer erfolgen. Nicht umsonst gehört zur beliebten Arduino-IDE standardmäßig eine „Servo"-Bibliothek. Aber wenn es tatsächlich um eine exakte Positionierung geht, werden meist Schrittmotoren verwendet.

Q *Wie geht das mit der Versorgungsspannung von Schrittmotoren? Auch Gleich- und Wechselspannung?*

A Die „Mechanismus" von Schrittmotoren ist grundsätzlich der gleiche wie bei Gleichstrommotoren, allerdings ist das Ganze wegen der unterschiedlichen Anwendung nicht mehr so einfach. Ein Gleichstrommotor dreht, wenn er versorgt wird und steht still, wenn er spannungslos ist. Ein Schrittmotor steht die meiste Zeit still, wird aber trotzdem mit Strom versorgt, damit er seine Position halten kann. Um bei einem stehenden Schrittmotor den Strom in Grenzen zu halten, ist der Widerstand des Stators relativ hoch. Dies unterscheidet ihn vom normalen Gleichstrommotor, wo der Rotorwiderstand aus Gründen des Wirkungsgrads besonders niedrig gehalten wird. Erhöht man bei einem Schrittmotor die Betriebsspannung, so kann besonders beim still stehenden Motor die Wärmeentwicklung zum Problem werden. Daher sollte man die angegebene Versorgungsspannung immer einhalten.

Q *Haben Schrittmotoren auch Nachteile?*

A Das häufigste Problem bei Schrittmotoren ist der „Verlust" von Schritten bei hoher Schrittfrequenz oder übermäßiger Last. Die Positionierung ist dann nicht mehr korrekt. Ein weiterer häufig auftretender Makel sind Resonanzen bei niedrigen Schrittfrequenzen, die als Brummen hörbar werden. Gravierender als das Brummen ist, dass Resonanzen das Getriebe beschädigen können. Daher dürfen auch keine „gewöhnlichen", mit Spiel behafteten Zahnräder im Getriebe verwendet werden. Ein spielfreier Zahnriemen ist die bessere Lösung!

Q *Gibt es eine Alternative zu Schrittmotoren?*

A Die Stärke von Schrittmotoren ist die exakte Positionierung ohne Feedback. Wenn Sie ein Feedback in Form eines Encoders oder eines Potis (zum Beispiel auf der Motorachse) nutzen, ist im Prinzip jeder Motor zur Positionierung geeignet (**Bild 5**).

Q *Sind Encoder nicht furchtbar teuer?*

A Nicht unbedingt. Wenn es nicht ungeheuer genau sein muss, funktioniert auch eine Lochscheibe mit Lichtschranke oder ein Punkt auf der sich drehenden Achse. So etwas kann man leicht selber bauen! ⏮

(160219)

Weblinks

https://de.wikipedia.org/wiki/Gleichstrommaschine

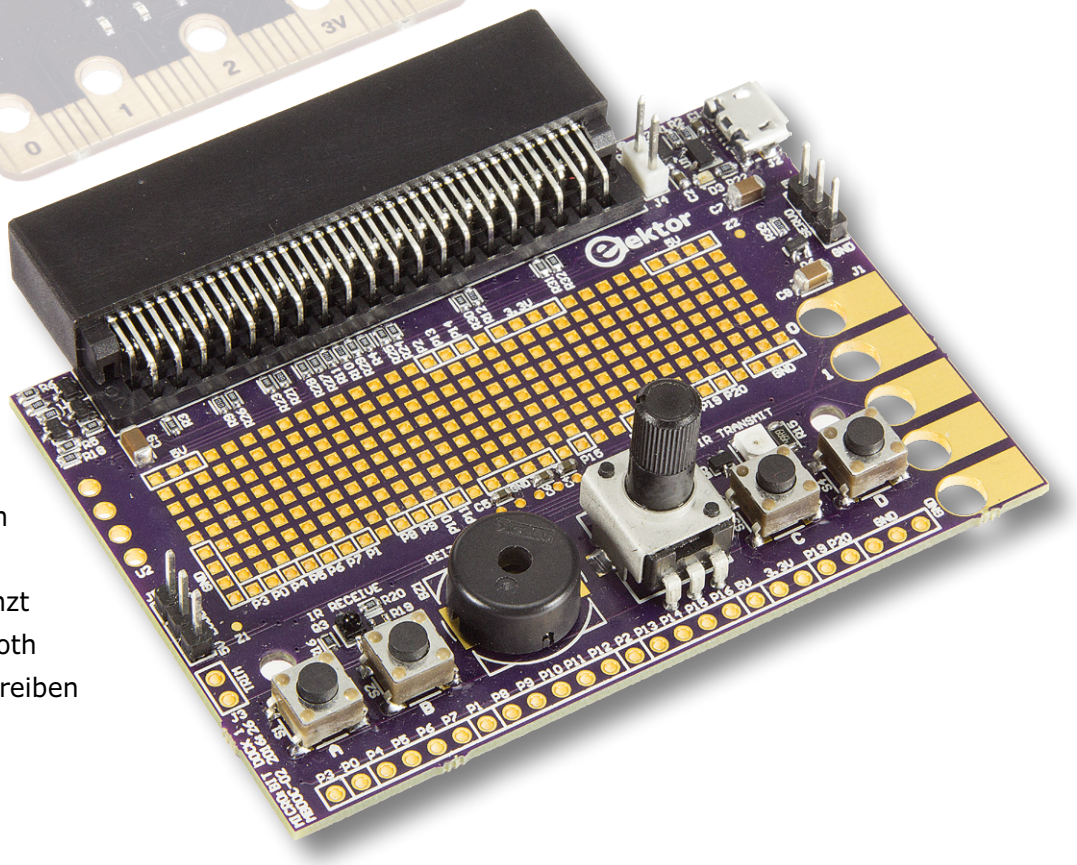
https://de.wikipedia.org/wiki/Wechselstrommotor

https://de.wikipedia.org/wiki/Schrittmotor

Dock für BBC micro:bit

„Basis an micro:bit, das Dock ist freigegeben!“

Von **Jason Long** (Kanada)



Mindestens eine Million BBC micro:bits treiben sich irgendwo in Großbritannien und dem Rest der Welt herum. Es gibt extrem viele Sachen, die man mit einem micro:bit machen kann. Es stimmt allerdings, dass sein User-Interface etwas begrenzt ist, wenn man keine über Bluetooth gekoppelte Smartphone-App schreiben mag. Hier kommt Abhilfe!

INFOS ZUM PROJEKT

- Mikrocontroller
- BBC micro:bit
- Bluetooth
- Prototyping

- Einsteiger
- ➡ Fortgeschrittener
- Experte

- etwa 2 Stunden
- Löten von SMDs
- etwa 20 €

Selbst viele junge (und ältere) Hacker wollen nicht zwingend auf ein Smartphone angewiesen sein. Außerdem gibt es viele Anwendungen, die prima mit einem micro:bit ohne Smartphone auskommen.

Grund genug für ein Board, das als eine Art Dock fungiert und das micro:bit mit vielen zusätzlichen Fähigkeiten und Funktionen nachrüstet. Das Ziel dieses ersten Dock-Projekts ist es, für mehr I/O-Pins zu sorgen und Platz für eine Prototypen-Fläche vorzusehen. Außerdem werden die kompletten Pins von micro:bit für ein Steckbrett kompatibel gemacht.

Der böse Edge Connector

Der sogenannte Edge Connector des micro:bit – zu Deutsch „Randleistenverbinder“ bzw. eine mit Kontaktflächen versehene Platinenkante – ist zwar preiswert in der Herstellung und sieht nett aus, macht es aber sehr schwierig, das Board irgendwo einzustecken. Die passenden Slot-Leisten sind nicht gerade billig und zudem nur selten mit den Lochabständen eines Steckbretts kompatibel. Eine erste Suche nach einer preiswerten Lösung für dieses Projekt ließ fast das ganze Vorhaben scheitern. Kleine Probleme können also große Auswirkungen haben. Doch ist dem Ingenieur bekanntlich nichts zu schwer: Die Recherche war schließlich doch noch erfolgreich und eine preiswerte Slot-Leiste wurde entdeckt. Wir waren erleichtert, denn so konnte das Projekt endlich Fahrt aufnehmen.

Aus der Erfahrung mit vielen Entwicklungs-Boards haben sich bestimmte Wünsche herauskristallisiert, die plus einem „Engenuics Daughter Board“ für die Verbindung mit unzähligen weiteren Boards in das Dock einfließen sollten. Erste Überlegungen waren eher theoretisch, denn ganz zu Anfang war es schwierig, überhaupt an ein micro:bit heranzukommen.

Zunächst nutzten wir eine zweistufige Anschlusslösung, als die preiswerte Slot-Leiste (**Bild 1**) noch nicht gefunden war. Bei einem kleinen Prototypen zeigte sich, dass die Bauteilekosten zu hoch waren und das Board noch zu groß war. Also wurde es bei strikter Beachtung der Kosten trotz Beibehaltung wichtiger Eigenschaften neu geplant.
Die zweite Lösung verzichtete auf hochpreisige Bauteile und besaß auch schon die neue, preiswerte Slot-Leiste. Da es noch einige Änderungen bei der Pin-Belegung der finalen Version des BBC micro:bit gab, wurden diese noch eingepflegt. Außerdem war das LCD nun optional und wurde direkt über dem Prototypenfeld angeordnet, um eine möglichst kleine Platine zu ermöglichen (**Bild 2**). Da nicht klar war, was sich in Zukunft noch alles beim micro:bit ändern würde, sind alle für die Dock-Hardware genutzten Pins durch Entfernen eines Jumper-Widerstands abtrennbar. Dies vermeidet, dass es bei unvorhergesehenen Änderungen zu unerwünschten Interaktionen mit dem Dock kommen kann.
Bei der Version für die Produktion gab es noch ein paar kleinere Änderungen im Layout und der Lötstoppmaske; aber dann war es soweit und die Sache konnte losgehen. In **Bild 3** können Sie das Resultat betrachten.

Mehr Taster

Die einfachste Form der Informationseingabe geschieht nach wie vor über Taster. Deswegen kann man davon eigentlich nie genug haben. Nach einigen Diskussionen schien eine Verdoppelung durch zwei weitere Taster (zu A & B kommen noch C & D hinzu) angebracht. Mit dem Dock stehen jetzt also insgesamt vier Taster zur Verfügung, die sich einfach per Software abfragen lassen.

Töne und Servos

LEDs stehen bei den preiswerten digitalen Indikatoren natürlich an erster Stelle, doch dicht dahinter kommt schon Sound. Mit einer einfachen PWM-Peripherie oder schlicht von einem malträtierten Bit gesteuert kann man sehr simpel Leute (und Hunde) mit schrägen Tönen in den Wahnsinn treiben. Auch ein Piezo-Buzzer produziert ja nicht unbedingt Wohlklänge, sondern das Gegenteil. Dennoch erweitert Sound jedes Projekt um eine weitere Dimension. An dieser Stelle daher die Bitte um Verzeihung an alle Lehrer, die in einem Klassenraum voller micro:bits zu unterrichten versuchen.

Außerdem gibt es noch einen Header zur Ansteuerung eines Servos. Die SERVO-Leitung erfährt dabei eine Pegelwandlung von den 3,3 V eines micro:bit auf notwendige 5 V, sofern die Sache per USB und nicht von Batterien versorgt wird. Die SERVO-Leitung ist mit IR_OUT gemultiplext. Wenn man beide Widerstände bestückt lässt, wird man beim Ansteuern eines Servos zwischendrin auch immer einige IR-Bits in den Raum blasen. Normalerweise wird ja auch nicht gleichzeitig ein Motor angesteuert und eine IR-Übertragung vorgenommen (vielleicht mit der einen Ausnahme von miteinander kämpfenden Laser-Robotern auf der Basis von micro:bits).

IR-Sender und -Empfänger

Ein micro:bit unterstützt zwar schon BLE (Bluetooth Low Energy), doch ist das nicht so einfach zu beherrschen. Dennoch ist eine Funkanbindung eine coole Sache! Das Dock vereinfacht eine Kommunikation durch einen zusätzlichen IR-Transceiver. Damit kann man mit wenig Aufwand z.B. Daten zwischen zwei Boards übertragen oder auch Komplexeres wie etwa IrDA einsetzen. Es gibt viele Anwendungen, welche die Magie des unsichtbaren Sendens und Empfangens von Information demonstrieren.

Analog

Ein Embedded-System ohne analoge Funktionen ist zwar möglich, aber meistens nutzlos. Wir leben ja immer noch in einer analogen Welt und nicht in einer Matrix (oder etwa nicht?). Das Dock ist mit einem 10-kΩ-Poti bestückt und so beschaltet, dass man damit jede Spannung zwischen den Leitungen zum Analogeingang einstellen kann. Die MCU eines micro:bit MCU hat einen 10-bit-ADC, mit dem man viele interessante Dinge anstellen kann.
Da der analoge Eingang auch mit Pin 1 verbunden ist, kann man das Poti abtrennen, damit hier keine ungewünschten Spannungen an dort angeschlossene andere Bauteile gelangen. Diese Anord-

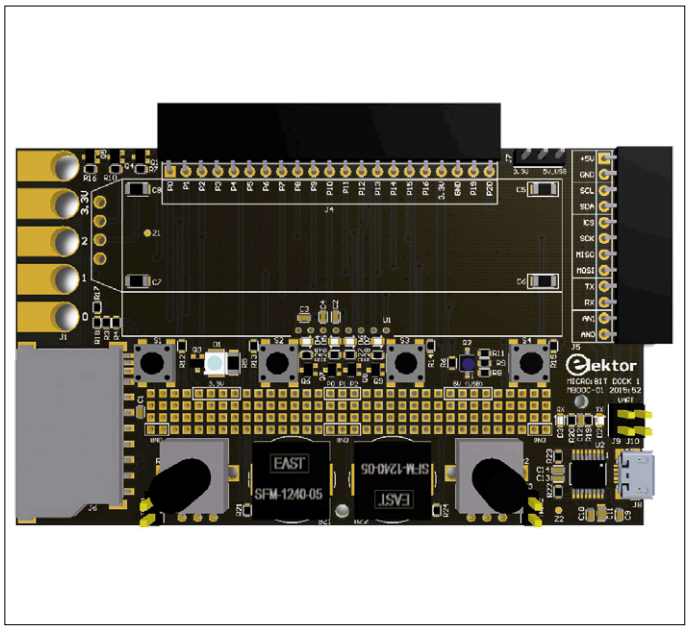


Bild 1. 3D-Rendering des ersten Entwurfs eines Docks für BBC micro:bit.

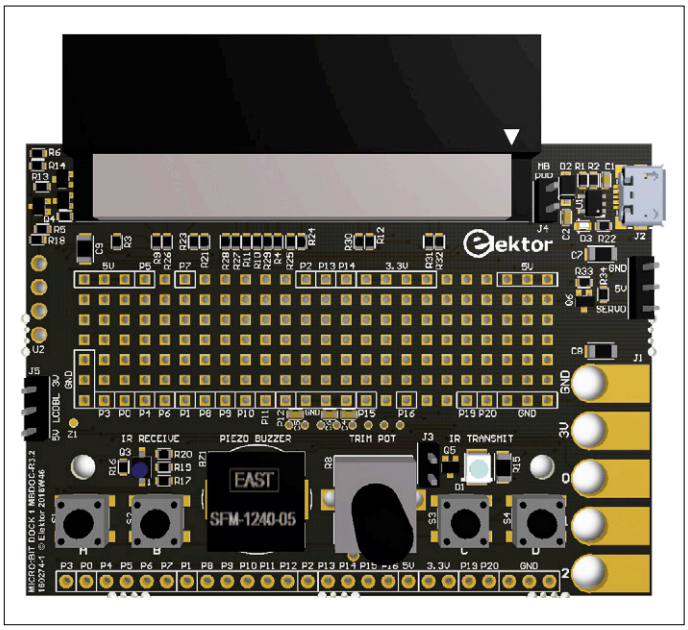


Bild 2. Die endgültige Version eines Docks für BBC micro:bit sieht so aus.

nung erlaubt jederzeit einen kurzen Test der Analogfunktionen eines Projekts, indem man den Schleifer des Potis kurz mit dem analogen Eingang verbindet.

Power

Der größte Teil der Elektronik des Docks kann zwar vom micro:bit mitversorgt werden, doch ist die Sache eigentlich zur Versorgung durch ein externes Netzteil konzipiert. Das Dock hat hierfür eine eigene Micro-USB-Buchse, dessen 5-V-Schiene man nutzen kann. Für die nötigen 3,3 V ist ein LDO-Regler vorgesehen. Mit Hilfe eines Jumpers kann man die Versorgung von Dock und micro:bit trennen. Zur Versorgung eignet sich jedes kleine 5-V-USB-Netzteil bzw. 5-V-Handy-Ladegerät mit passendem Kabel.

LCD (optional)

Den größten Komfort bietet immer noch ein „richtiges" Display, weshalb ein LCD zur optionalen Bestückung vorgesehen

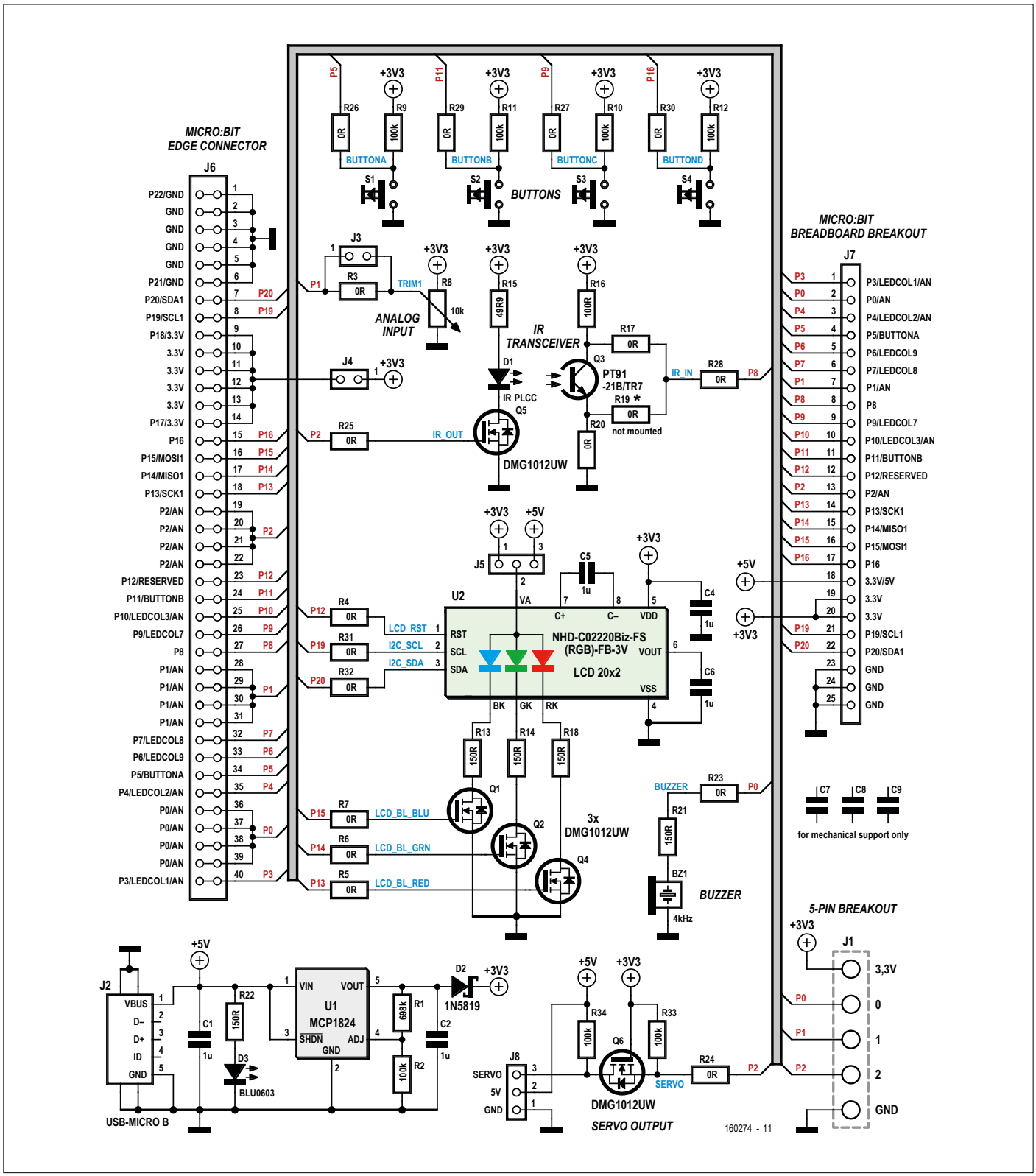
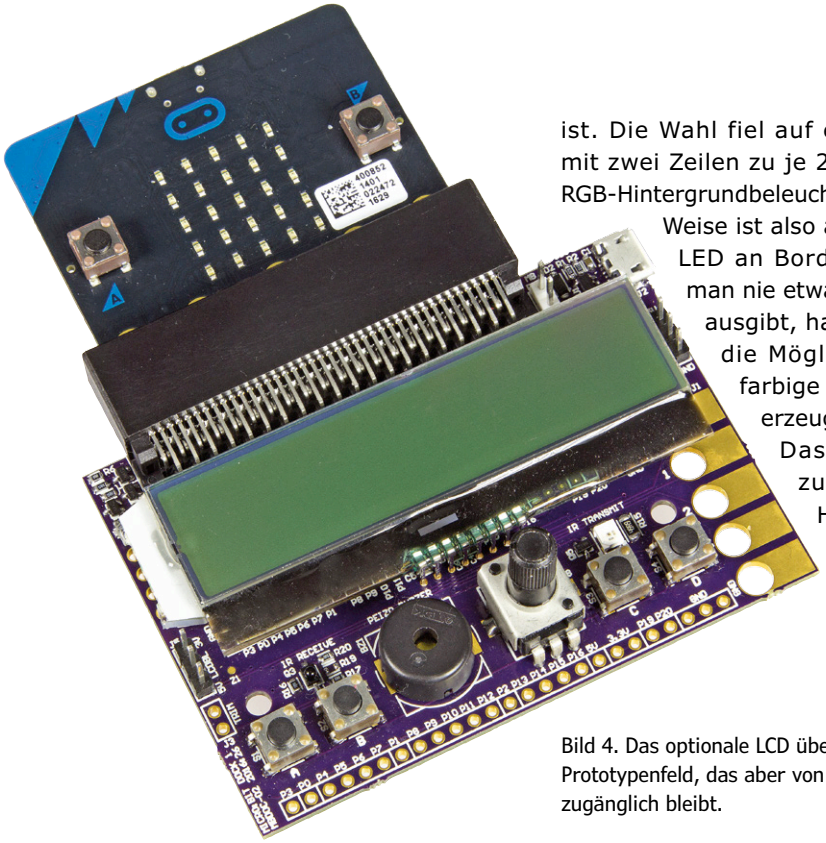


Bild 3. Der Schaltplan des Docks.



ist. Die Wahl fiel auf ein ASCII-LCD mit zwei Zeilen zu je 20 Zeichen und RGB-Hintergrundbeleuchtung. Auf diese Weise ist also auch eine RGB-LED an Bord. Selbst wenn man nie etwas auf dem LCD ausgibt, hat man so doch die Möglichkeit, schön farbige Lichteffekte zu erzeugen.

Das LCD ist zwar zum klassischen HD44780 kompatibel, nutzt aber ein I²C-Interface und spart so viele Pins. Via Jumper J5 wird das RGB-Licht entweder von der 3,3- oder 5-V-Schiene versorgt. Die manuelle Einstellung erspart Code und beschreibenden Text.

Bild 4. Das optionale LCD überdeckt das Prototypenfeld, das aber von der anderen Seite zugänglich bleibt.

Prototypenfeld

Auch hiervon kann man nie genug haben. Auf der Dock-Platine befindet sich ein Lochraster-Feld, auf dem man eigene Schaltungen aufbauen kann. Die Pads sind so geformt, dass man nicht nur bedrahtete Bauelemente, sondern auch SMDs im Format 0603 bestücken kann. Alle Pins des micro:bit stehen mitsamt Stromversorgung und Masse zur Verfügung. Beide Platinenseiten sind mit Labeln versehen. Selbst wenn eine Seite mit einem LCD abgedeckt ist, kann man doch noch von der anderen Seite Bauteile bestücken. Auch die Anschlüsse für die Löcher 0, 1, 2, GND und 3,3 V des micro:bit sind

Tabelle 1. Pin-Belegung von BBC micro:bit und Dock.

micro:bit Pin	Edge Connector Pins	Funktionen	Dock (main)	Dock (alt)	Kommentar
3,3 V	10-13	3,3 V	3,3 V		J4 verbindet micro:bit 3,3 V mit Dock 3,3 V
GND	2-5	GND	GND		
P0	36-39	P0/AN	BUZZER	P0 Access	Trennen mit R23
P1	28-31	P1/AN	TRIM1	P1 Access	Trennen mit Jumper J3 oder R3
P2	19-22	P2/AN	SERVO	IR_OUT	SERVO trennen mit R24; IR_OUT trennen mit R25
P3	40	P3/LEDCOL1/AN			
P4	35	P4/LEDCOL2/AN			
P5	34	P5/BUTTONA	BUTTONA		Trennen mit R26
P6	33	P6/LEDCOL9			
P7	32	P7/LEDCOL8			
P8	27	P8	BUTTONC	IR_IN	BUTTONC trennen mit R27; IR_IN trennen mit R28
P9	26	P9/LEDCOL7			
P10	25	P10/LEDCOL3/AN			
P11	25	P11/BUTTONB	BUTTONB		Trennen mit R29
P12	23	P12/RESERVED	LCD_RST*		Trennen mit R4
P13	18	P13/SCK1	LCD_BL_RED*		Trennen mit R5
P14	17	P14/MISO1	LCD_BL_GRN*		Trennen mit R6
P15	16	P15/MOSI1	LCD_BL_BLU*		Trennen mit R7
P16	15	P16	BUTTOND		Trennen mit R30
P17	14	P17/3.3V	3.3V		
P18	9	P18/3.3V	3.3V		
P19	8	P19/SCL1	LCD SCL*		Trennen mit R31
P20	7	P20/SDA1	LCD SDA*		Trennen mit R32
P21	6	P21/GND	GND		
P22	1	P22/GND	GND		

* Nur mit optionalem LCD.

vorhanden, da sie nicht mehr zugänglich sind, wenn man das micro:bit eingesteckt hat. Das LCD überdeckt teilweise die Leitungen für 3,3 V und GND (**Bild 4**), aber ein Clip passt da immer noch. Die drei restlichen I/O-Leitungen sind voll für Bananenstecker zugänglich. Selbstverständlich liegen alle Pins von micro:bit auf einer normalen Stiftleiste mit dem Rastermaß 0,1“ auf der Rückseite des Docks. Man kann dort einen Stecker nach Wahl anschließen oder das Dock auf ein Steckbrett stecken. Die Pin-Anordnung des Edge Connectors blieb erhalten und wurde durch einen 5-V-Pin ergänzt.

Zusammenbau

Nachdem alle Eigenschaften festgelegt wurden, war es nicht mehr sehr weit zu einem detaillierten Schaltplan und von da aus zur Platine. Wichtig war die Einhaltung der Pin-Anordnung (**siehe Tabelle 1**) und das Heraushalten aus der LED-Matrix, da schon die Zweckentfremdung einer Leitung zur Störung aller LEDs geführt hätte. Das ließ nicht viele Freiheiten, weshalb dies bei einigen Funktionen zu Doppelungen führte.

Welche Projekte?

Da micro:bit (**Bild 5**) wirklich Spaß macht, war der Code für einige Beispiel-Projekte schnell geschrieben.

STÜCKLISTE

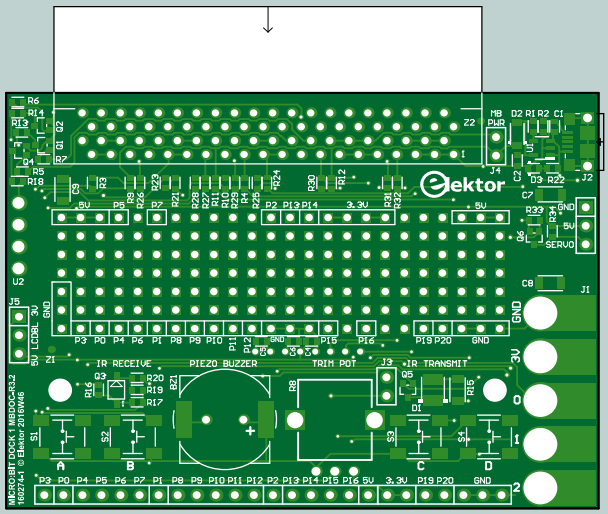
Widerstände:
(1 %, 0,1 W, SMD 0603)
R3..R7,R17,R20,R23..R32 = 0 Ω
R19 = 0 Ω, nicht bestückt
R16 = 100 Ω
R2,R9..R12,R33,R34 = 100 k
R13,R14,R18,R21,R22 = 150 Ω
R15 = 49,9 Ω (SMD 1206)
R1 = 698 k
R8 = 10-k-Poti, linear

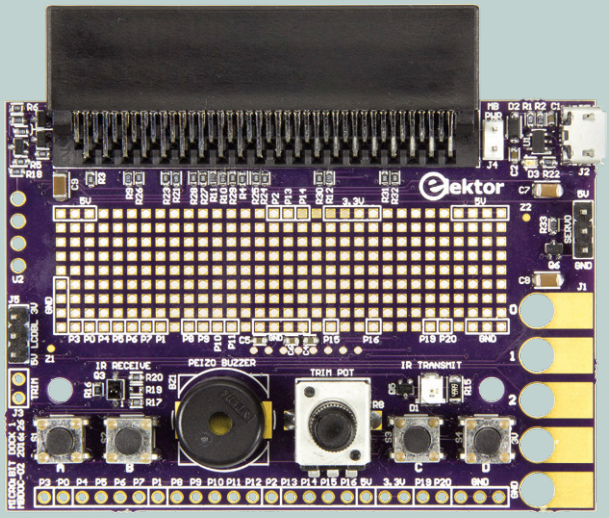
Kondensatoren:
C1, C2, C4, C5, C6 = 1 µF (0603)
C7,C8,C9 = jeder Wert, SMD 1206, Höhe 0,85 mm

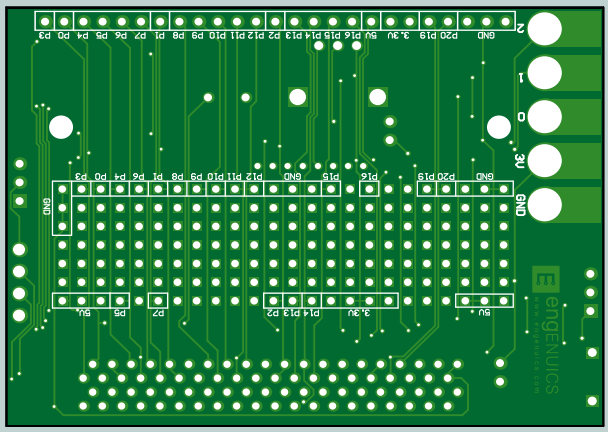
Halbleiter:
D2 = 1N5819HW-7-F
D1 = IR67-21C/TR8 (IR-Diode)
U1 = MCP1824-ADJE/OT
U2 = LCD NHD-C0220BiZ-FS(RGB)-FB-3V NNP (optional)
D3 = LED, blau, SMD 0603
Q1,Q2,Q4,Q5,Q6 = DMG1012UW
Q3 = PT91-21B/TR7

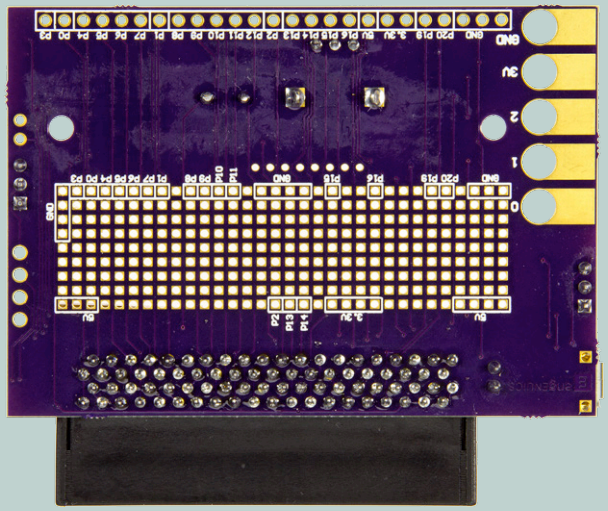
Außerdem:
J6 = Edge Connector, 2x40-pol., RM 1/20”
J7 = 25-pol. Stiftleiste, RM 0,1”
J3,J4 = 2-pol. Stiftleiste, RM 0,1”
J5,J8 = 3-pol. Stiftleiste, RM 0,1”
J2 = USB-Buchse Micro B
BZ1 = Piezo-Buzzer
S1,S2,S3,S4 = Taster, 6x6 mm

Platine 160274-1 rev3.2









An den folgenden Ideen wird noch gearbeitet:

- Laser-Tag: Mit den IR-Sende- und Empfangsfunktionen lassen sich andere „wegblasen“. Dank vieler Optionen kann man die Reichweite der relativ schwachen IR-LED austesten.
- IR-Alarm: Raumüberwachung mit unsichtbarem IR-Licht, das einen Zähler triggert oder einen Alarm auslöst, wenn der Strahl unterbrochen wird.
- Lichtsteuerung: Mit Hilfe des Poti über BLE das Licht im Raum dimmen.
- Einfacher Text-Messenger (mit LCD): Dank der einfachen Möglichkeiten ist die Eingabe von Text zwar mühevoll, aber er lässt sich gut von einem zu einem anderen Board oder zu einem Smartphone übertragen. Man kann auch Standard-Nachrichten programmieren und sie mit einem Tastendruck verschicken.
- Stoppuhr (optional LCD): Eine Stoppuhr mit vielen Fähigkeiten bauen.
- Nerv-Buzzer 1: Mit dem Poti beliebige Töne einstellen und via Buzzer ausgeben. Verfolgt man mit einem Oszilloskop die Form des PWM-Signals, wird schnell klar, wie eine digitale Audio-Erzeugung funktioniert.
- Nerv-Buzzer 2: Einen hohen Ton einstellen und das Board unter dem Schreibtisch eines Freundes platzieren. Per Funk einschalten und wieder deaktivieren, wenn er dem Board nahekommt.
- IR-Fernbedienung: Mit micro:bit und Dock die wichtigsten Funktionen des Fernsehers fernsteuern (oder einfach eine IR-Verbindung zwischen zwei Boards herstellen).

Wenn sie noch weitere Ideen haben, lassen Sie uns das wissen. Kommentare und Vorschläge sind explizit erwünscht. Gutes Andocken!

(160274)

EINKAUFSZETTEL

→ 160274-91
fertig aufgebautes Board

→ SKU 17773
BBC micro:bit

→ SKU 17589
5V AC/DC 2A Netzteil mit MicroUSB-Anschluss oder

→ SKU 18012
Batteriehalter 2x AA

Weblinks

[1] www.elektormagazine.de/160274

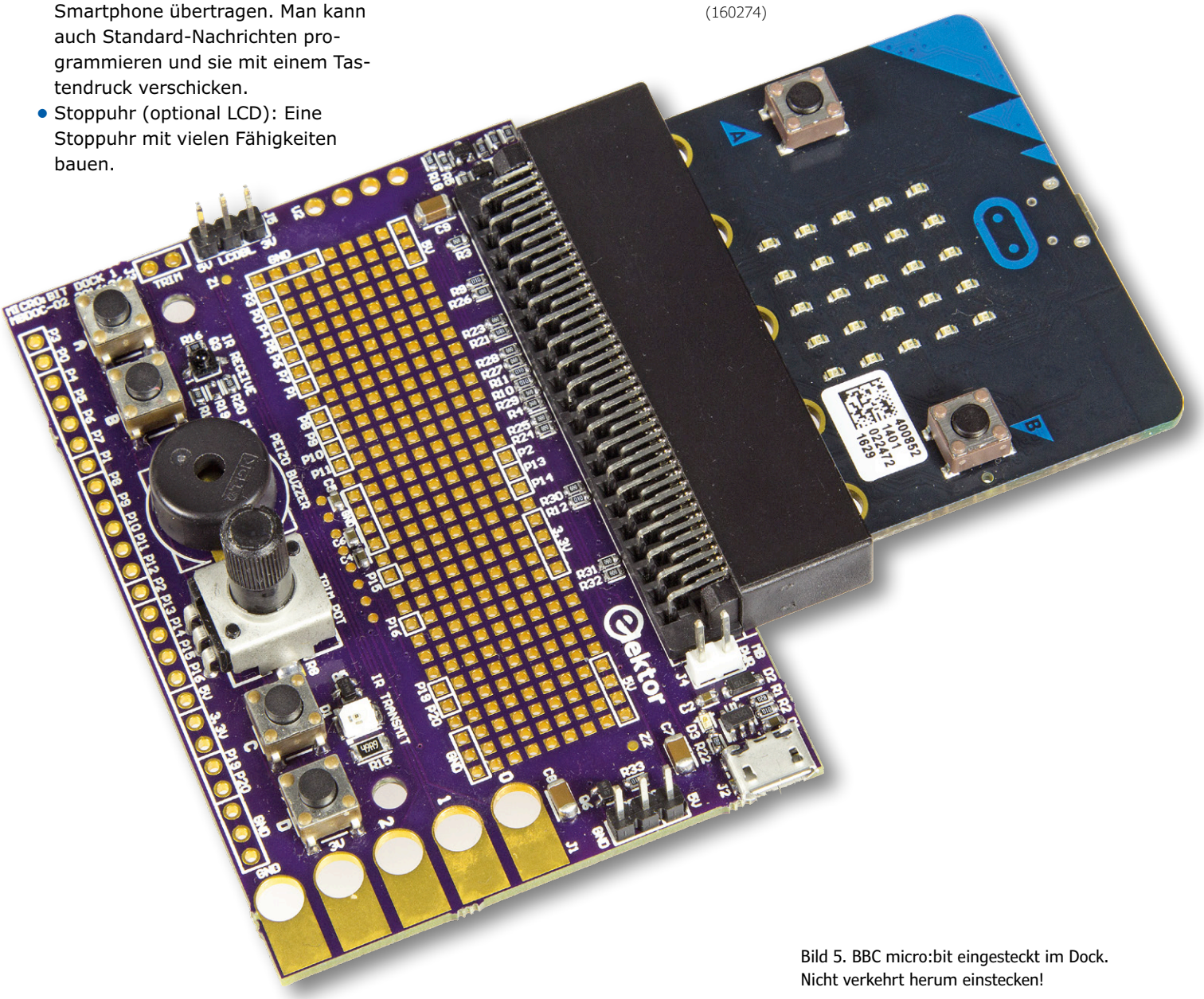


Bild 5. BBC micro:bit eingesteckt im Dock. Nicht verkehrt herum einstecken!

Magnetblasenspeicher

Bemerkenswerte Bauteile

Von **Neil Gruending** (Kanada)

Wir haben schon über Ringkerne als Speicherelement gesprochen, die zu ihrer Zeit zwar gut, aber auch ziemlich groß und ziemlich teuer waren. Einer der interessanten Nachfolger waren die Magnetblasenspeicher, die Daten in winzigen magnetisierten „Blasen" speicherten. Viele Leute dachten damals, es wäre das ultimative Gedächtnis, das alle anderen Speichertypen für immer ersetzen würde. Und für eine Weile sah es auch so aus. Ringkernspeicher waren schwierig herzustellen, weil sie manuell gebaut werden mussten. Dies veranlasste Leute wie Andrew Bobeck bei den Bell Laboratories, nach Alternativen zu suchen. So entstand der Twistor-Speicher, der ein Magnetband anstelle der Ringkerne verwendete und billiger zu produzieren war. Wo ein Strom durch die Drähte in dem Band floss, verschoben sich die magnetischen Felder in Flussrichtung des Stroms. Dieses Verhalten stellt die Grundlage des Magnetblasenspeichers dar. Auf dem Band waren die Bits in relativ großen magnetischen Bereichen gespeichert, um das Feld – die so genannte Domäne - stabil zu halten. Bobeck entdeckte, dass diese Bereiche durch Einwirken eines Magnetfeldes auf das Substratmaterial zu kleinen Kreisen oder, wie er sagte, Blasen verkleinert werden konnten. Durch die Lektüre von Paul Michaelis Arbeit über Substrate war es ihm möglich, die Blasen gezielt durch ein Substrat aus einem Silikat (Granat) zu bewegen. Damit war der Blasenspeicher geboren.

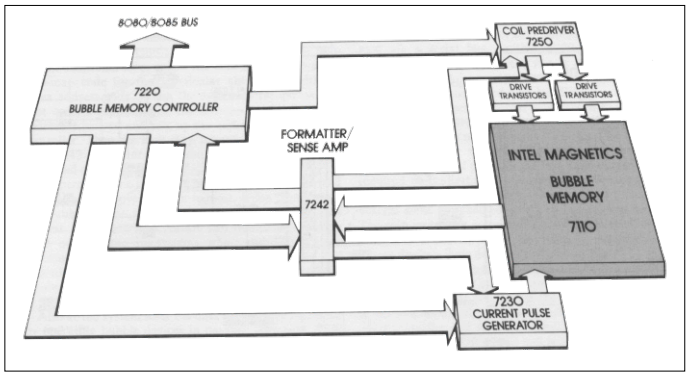


Bild 1. Blockdiagramm des Intel 128 KB Bubble Memory [1].

Bild 1 zeigt den Aufbau eines frühen Blasenspeichers (Intel 7110 128 KB). Der eigentliche Teil mit dem Blasenspeicher wurde als 2D-Array auf dem Substrat angeordnet. Eine Seite des Arrays beschreibt die Speicherbits, die andere Seite liest sie aus. Die Bits werden durch kleine Elektromagnete von der Schreib- zur Leseseite bewegt. Die Anzahl der Bits, die wie in einem gigantischen Schieberegister zwischen beiden Enden gespeichert werden konnte, definierte die Speicherkapazität. Die Magnetblasenspeicher wurden meist mit überdimensionierter Kapazität hergestellt, so dass ausgefallene Speicherbereiche ausgeblendet werden konnten. Diese Maskenwerte wurden dann entweder auf dem Etikett (wie beim TI TBM0103

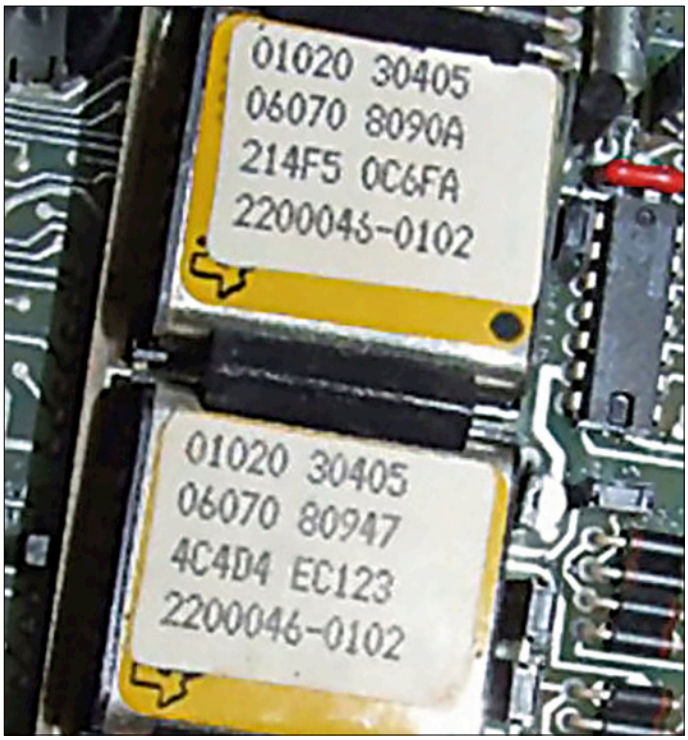


Bild 2. Blasenspeicher von TI [2].

in **Bild 2**) angegeben oder intern gespeichert, so dass der Speichercontroller auf den Wert zugreifen konnte.

Magnetblasenspeicher waren auf dem besten Weg, zum sogenannten *universal memory* zu werden, weil sie nicht flüchtig waren und einen direkten Zugriff erlaubten. Fast alle Halbleiterhersteller forschten in diese Richtung, doch wurden die Magnetblasenspeicher bald von verbesserten Festplatten und RAM-Speichern ersetzt. Nur in (militärischen) Nischenanwendungen, in denen es auf besondere Zuverlässigkeit in rauen Umgebungen (im Vergleich zur Festplatte der 80er Jahre) ankam, blieben Magnetblasenspeicher, die ja keine beweglichen Teile besaßen, noch eine Zeitlang erhalten. Doch schließlich wurden sie auch hier von den aufkommenden Flash-Speichern abgelöst.

Im Internet gibt es viele Informationen über Magnetblasenspeicher. Gebrauchte Bauteile sind gut verfügbar. Vielleicht möchten Sie es mit dieser Technologie ja einmal versuchen, wenn Sie beim nächsten Mal einen nichtflüchtigen Speicher für Ihr Projekt benötigen. ⏮

(160215)

Weblinks

[1] Intel Bubble Memory Design Handbook: https://ia802606.us.archive.org/18/items/IntelBubbleMemoryDesignHandbook/BubbleMemoryDesignHandbook.pdf

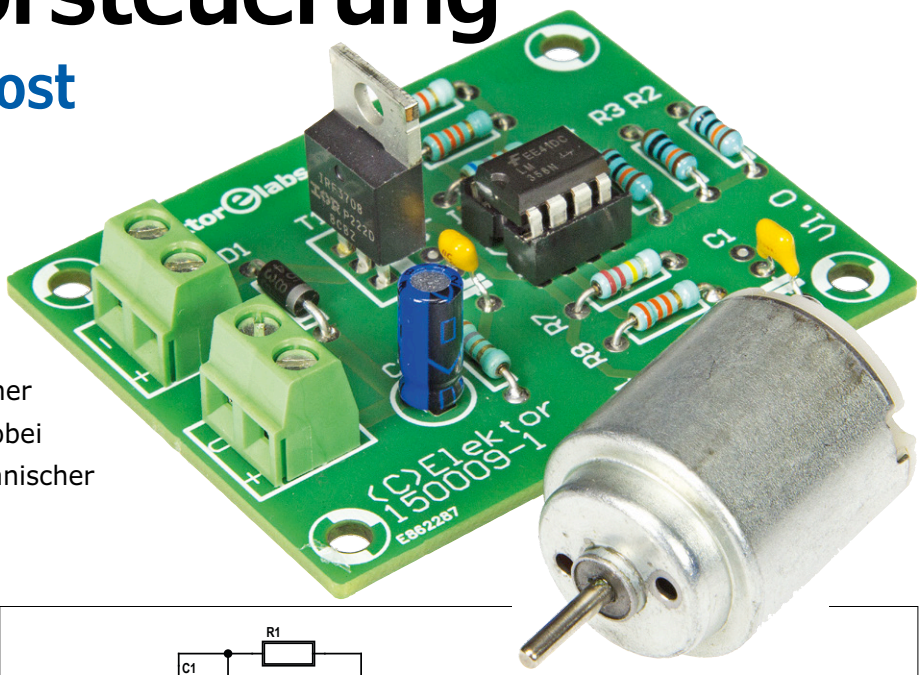
[2] www.wylie.org.uk/technology/computer/bubblmem/bubblmem.htm

PWM-Motorsteuerung

Mit Duty-cycle-Boost

Von **Rolf Blijleven** (NL)

Ein Gleichstrom-Kleinmotor wird hier so gesteuert, dass er mit niedriger, nahezu lastunabhängiger Drehzahl läuft. Möglich ist dies mit einer pulsbreitenmodulierten Spannung, wobei der Duty-cycle bei wachsender mechanischer Last angehoben wird.



Projekte mit Gleichstrom-Motoren erfordern gelegentlich, dass sich Motoren sehr langsam drehen. Das lässt sich bewerkstelligen, indem der Motor an einer rechteckförmigen Spannung betrieben wird, deren Duty-cycle nur wenige Prozent beträgt. Doch wenn die mechanische Last wächst, besteht die Gefahr, dass der Motor abrupt stehenbleibt. Das Problem kann mit einer pulsbreitenmodulierten Motor-Betriebsspannung (PWM) gelöst werden, wobei der Duty-cycle variabel ist.

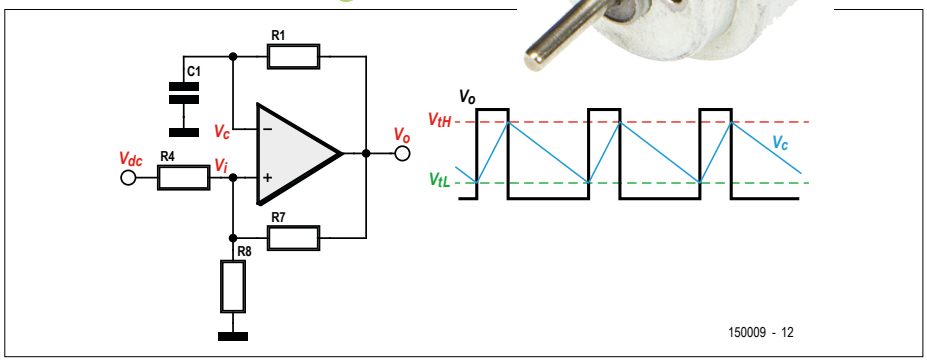


Bild 1. Prinzipschaltung eines Rechteckgenerators mit nur einem Opamp.

INFOS ZUM PROJEKT

Hobby
Motor PWM
Duty-cycle

➡ Einsteiger
Fortgeschrittener
Experte

Etwa 0,5 Stunden

Motor aus CD/DVD-Laufwerk

Rund 15 Euro

Ein von Hand variierbarer Duty-cycle stellt noch keine befriedigende Lösung dar, sinnvoller ist erst die selbsttätige Anpassung an die Last. Wir zeigen hier, wie sich eine solche Steuerung ohne Mikrocontroller realisieren lässt. Die Steuerung misst den Motorstrom, und ein daraus abgeleitetes Signal wird zum PWM-Generator rückgeführt. Zwei Opamps und ein MOSFET, einige Widerstände und Kondensatoren, das genügt, um eine solche Motorregelung aufzubauen. Basis ist die Prinzipschaltung in **Bild 1**, sie entstammt dem Datenblatt des Opamps LM358 [1]. Die Nummerierung der Bauelemente wurde an die definitive Schaltung in **Bild 2** angepasst.

Prinzip

Die Arbeitsweise dürfte nicht unbekannt sein: V_{dc} ist eine Gleichspannung (dc von *direct current*, nicht zu verwechseln mit DC für *Duty Cycle* [2]). Unmittelbar nach dem Einschalten ist die Spannung V_i am nicht invertierenden Opamp-Eingang größer Null. Die Spannung V_c am invertierenden Eingang ist noch Null, da der Kondensator nicht geladen ist. Die Spannung V_o am Opamp-Ausgang geht gegen die Betriebsspannung, sie ist jetzt V_{omax}. Diese Spannung lädt über Widerstand R1 so lange den Kondensator C1, bis V_c größer V_i ist. Die Spannung, bei der dies geschieht, heißt V_{tH}. Ausgangsspannung V_o kippt auf Null, so dass sich C1 über R1 auf Spannung V_{tL} entlädt. Bei V_{tL} liegt V_c unter V_i. Das hat zur Folge, dass V_o auf V_{omax} zurückkippt und der Zyklus von vorn beginnt. Die Schaltung ist eine Kombination aus einem Sägezahn-Oszillator und einem so genannten Schmitt-Trigger.

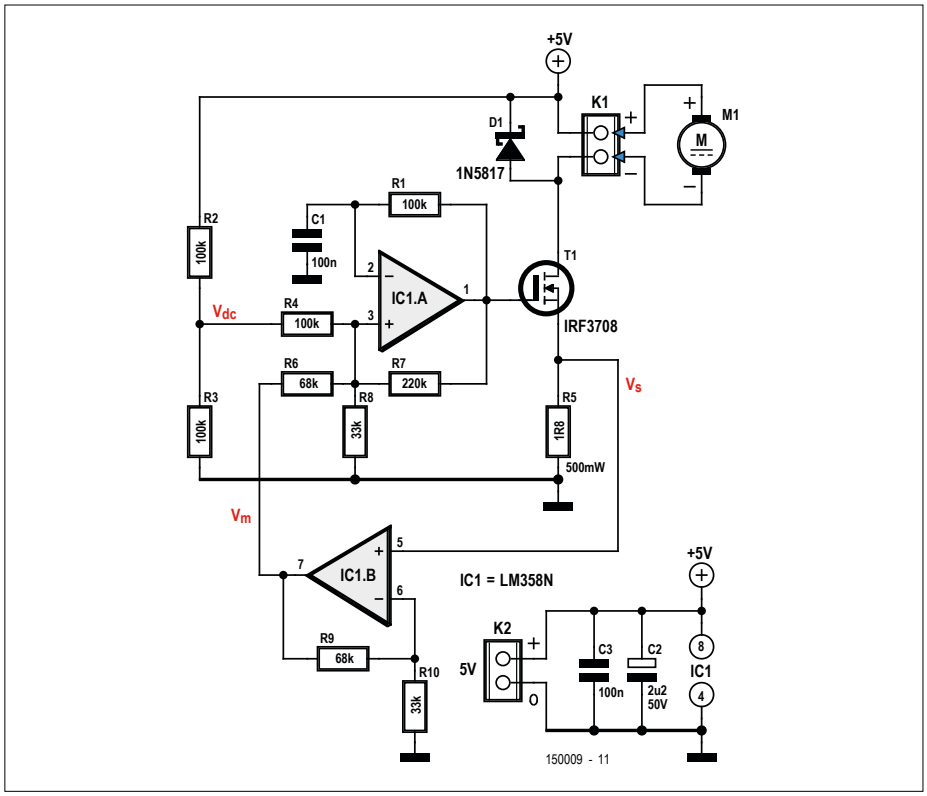


Bild 2. Der Rechteckgenerator IC1A steuert über MOSFET T1 den DC-Motor, während der Opamp IC1B mit Stromsensor R5 eine Mitkopplung bewirkt.

Mit viel Mathematik lässt sich berechnen, welche Werte die beteiligten Bauelemente haben müssen, um eine bestimmte Zyklus-Zeit und einen bestimmten Duty-cycle zu realisieren. Das Ergebnis ist aber näherungsweise auch intuitiv erfassbar. Mit einem beliebigen Opamp nach dem Vorbild des 741 liegt die Obergrenze von V_{omax} bei 1,5 V unter der Betriebsspannung. Soll die Obergrenze sehr nahe bei der Betriebsspannung liegen, ist ein Opamp mit FET-Ausgang die richtige Wahl, beispielsweise der CA3130 oder CA3140.

Bei hohen Werten von V_{dc} hat auch V_i hohe Werte, und auch der Duty-cycle ist hoch. Solange V_i nicht unter V_c sinkt, beträgt der Duty-cycle 100 %, am Opamp-Ausgang liegt eine Gleichspannung. Die wichtigste Variable ist hier die Spannung V_i.

Vielleicht werden die Zusammenhänge noch deutlicher, wenn die Überlegungen nicht mit Widerständen, sondern mit Leitwerten angestellt werden: Falls R4 die Spannung V_{dc} nur wenig in Richtung V_i leitet, trägt V_{dc} nur wenig zu V_i bei, und der Duty-cycle ist niedrig. Wenn R8 die Spannung V_i stark nach Masse ableitet, ist der Duty-cycle ebenfalls niedrig. Von R7 hängt der Einfluss der Spannung V_{omax} auf V_{tH} ab. Wenn R7 gut leitet, ist die Differenz zwischen V_{tH} und V_{VtL} hoch. Für den Duty-cycle ist jedoch die *Summe* aus V_{tH} und V_{tL} entscheidend, auf sie hat R7 weniger Einfluss als R4 und R8.

Nach diesen Überlegungen können R4, R7 und R8 so dimensioniert werden, dass der Duty-cycle stets für das Drehen des Motors ausreicht. Das Ziel ist jedoch, einen möglichst niedrigen Duty-cycle zu realisieren, der selbsttätig steigt, wenn der Motor mechanisch stärker belastet wird.

Schaltung

Das Ergebnis der Überlegungen zeigt die Schaltung in **Bild 2**. Spannungsteiler R3/R4 stellt V_{dc} auf die halbe Betriebsspannung ein. Das PWM-Signal am Ausgang von IC1A wird unmittelbar dem MOSFET zugeführt, der den Motor steuert. Mit R5, Wert 1,8 Ω, wird der Motorstrom gemessen, er beträgt nominal 450 mA. Daraus resultiert eine Spannung V_s von 0,8 V bei mittlerer Last, bei hoher Last steigt sie auf 1,2 V. In diesem Fall fließt durch R5 ein Strom von etwa 700 mA. Da Spannung und Strom pulsförmig sind, wird die von R8 in Wärme umgesetzte Leistung $I^2 \cdot R = 0{,}7^2 \cdot 1{,}8 = 0{,}9$ W um den Duty-cycle reduziert, er liegt in diesem Fall bei 20 %. Der Widerstand R5, mit den Werten 1,8 Ω und 0,5 W, wird deshalb kaum warm.

Die Spannung V_s wird von IC1B verstärkt, von dort gelangt die Spannung über R6 nach V_i. Dadurch steigt V_i umso mehr, je höher der Motorstrom steigt. Das ist der Fall, wenn der Motor mechanisch höher belastet wird.

Im Prinzip klingt dies einfach, doch die Dimensionierung der Widerstände bedarf

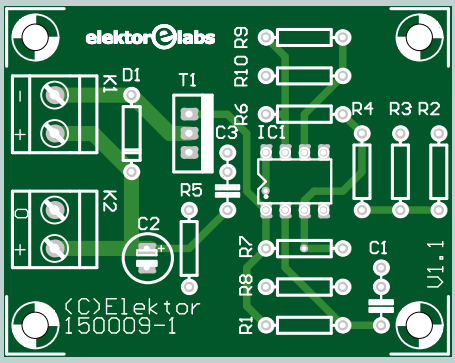
STÜCKLISTE

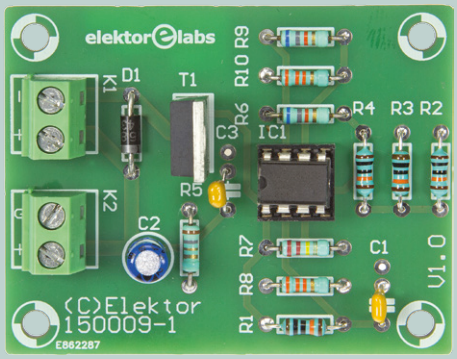
Widerstände: (5 %/0,25 W, wenn nicht anders angegeben)
R1,R2,R3,R4 = 100 k
R5 = 1Ω8, 0,5 W
R6,R9 = 68 k
R7 = 220 k
R8,R10 = 33 k

Kondensatoren:
C1,C3 = 100 n
C2 = 2µ2/16 V, RM 2 mm

Halbleiter:
D1 = 1N5817
T1 = IRF3708PBF
IC1 = LM358

Außerdem:
K1,K2 = Kabelschraubklemme 2-polig, RM 5,08 mm
DC-Motor mit Getriebe, ausgebaut aus CD/DVD-Laufwerk
Platine 150009-1





weiterer Überlegungen. Zum einen sinkt die Frequenz des PWM-Signals, wenn der Duty-cycle steigt, was an der mathematischen Beziehung deutlich wird. Entscheidend sind die Ladezeit t1 und die Entladezeit t2 des Kondensators C1:

$$Duty\text{-}cycle = \frac{t1}{t1+t2} \cdot 100\%$$

Wenn die Ladezeit steigt, steigt auch der Duty-cycle. Da aber die Frequenz 1/(t1+t2) beträgt, sinkt deren Wert. Die Anzahl der Impulse in der Sekunde nimmt niedrigere Werte an, so dass auch die Motordrehzahl sinkt. Das Ziel war jedoch, die Motordrehzahl konstant zu halten. Wenn V_s mehr Einfluss erhält, indem der Wert von R6 herabgesetzt wird, steigt zwar der Duty-cycle, doch die Drehzahl sinkt noch stärker.

Im Versuchsaufbau erwies sich eine niedrigere Zeitkonstante R1 · C1 nicht als die Lösung, denn der Motor gab bei PWM-Frequenzen über 75 Hz befremdliche Geräusche von sich, und über 120 Hz verweigerte er den Dienst.

Hinzu kommt, dass die Spannung an R5 eine Gleichspannung mit überlagerter Welligkeit ist. Die Welligkeit wird durch die Bürsten des Motors verursacht, wenn sich der Motor dreht. Bei hoher Motorlast geht die Spannung an R5 gegen einen Maximalwert. Dann sinkt die Frequenz der Welligkeit, denn der Motor dreht langsamer.

Der Motorstrom, umgesetzt in die verstärkte Spannung V_m, wird zur Spannung am *nicht invertierenden* Eingang des PWM-Opamps addiert. Das bedeutet, dass es sich hier um eine *Mitkopplung* handelt. Theoretisch ist dies keine gute Idee, unter anderem weil die Impedanz der Spannungsquelle die Addition beeinflusst. Doch das hat hier nur wenig Gewicht. Die Ausgangsimpedanz von IC1B ist erheblich niedriger als der Wert von R6. Die Quellenimpedanz von Spannungsteiler R2/R3 beträgt 50 kΩ, der halbe Wert von R4. Das Nachschalten eines Emitterfolgers zum Herabsetzen der Impedanz hat keine spürbare Wirkung.

Während die Prinzipschaltung in Bild 1 mit vergleichsweise unkomplizierter Mathematik erfassbar ist, traten bei der Schaltung in Bild 2 Phänomene auf, die sich dem widersetzten. Deshalb wurde diese Schaltung weitgehend auf experimentellem Weg entwickelt. Mit den dort und in der Stückliste angegebenen Werten betrug der Duty-cycle etwa 17 % bei der PWM-Frequenz 55 Hz, steigend bis 21 Hz und 45 Hz bei höchster mechanischer Last.

Das Elektor-Labor hat für diese PWM-Motorsteuerung eine kleine Platine entworfen, **Bild 3** zeigt das Layout. Damit ist die Schaltung zwar schnell aufgebaut, doch die kritischen Widerstände müssen vermutlich abhängig vom Motor-Typ neu dimensioniert werden. Beim nächsten Entwurf dieser Art, wenn die Drehzahl mit hoher Genauigkeit konstant bleiben soll, wird der Einsatz eines Mikrocontrollers nicht vermeidbar sein. ⏮

(150009)gd

Weblinks

[1] Datenblatt LM358: www.ti.com/lit/ds/symlink/lm158-n.pdf

[2] https://de.wikipedia.org/wiki/Tastgrad

EINKAUFSZETTEL

→ Platine
150009-1

Anzeige

You CAN get it...

Hardware und Software für CAN-Bus-Anwendungen

embeddedworld2017 Exhibition&Conference ...it's a smarter world

Besuchen Sie uns in Halle 1, Stand 483

PCAN-USB

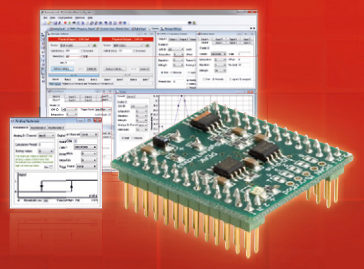
CAN-Adapter für den USB-Port. Optional auch mit galvanischer Trennung erhältlich.

ab 180 €

PCAN-RS-232

Programmierbarer Umsetzer für RS-232 auf CAN. Library und Programmierbeispiele wie die LAWICEL-Protokoll-Unterstützung im Lieferumfang enthalten.

110 €



PCAN-MicroMod

I/O-Modul mit CAN-Interface inkl. Library zur Entwicklung eigener Applikationen und Konfigurationssoftware für Windows.

98 €

Alle Preise verstehen sich zzgl. MwSt., Porto und Verpackung. Irrtümer und technische Änderungen vorbehalten.

www.peak-system.com

PEAK System

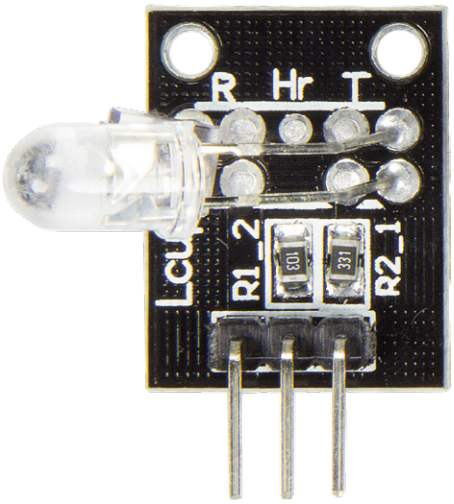
Otto-Röhm-Str. 69
D-64293 Darmstadt
Tel.: +49 6151 8173-20
Fax: +49 6151 8173-29
info@peak-system.com

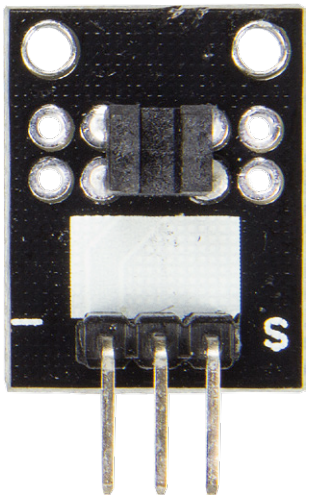
Sensoren (4)

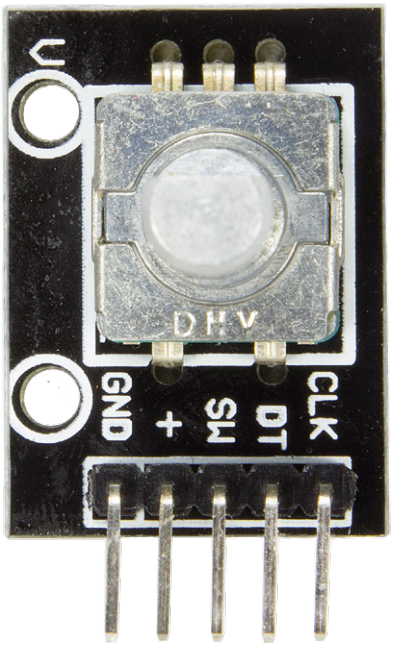
Für Arduino und Co.

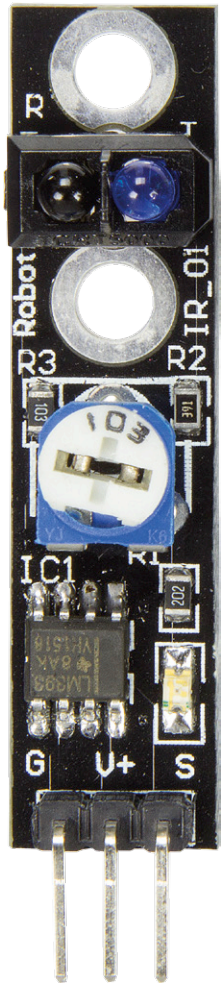
Schalten und Steuern mit Licht, das ist eine weit verbreitete Technik mit vielen Vorteilen. Keine mechanische Berührung, keine Abnutzung von Schaltkontakten, elektrisch isolierte Baugruppen und vieles mehr.

Von **Burkhard Kainka**









Hier sollen Lichtschranken und ihre Anwendungen vorgestellt werden, von der Gabellichtschranke bis zur Reflexlichtschranke und zum Puls-Detektor. Und es soll gezeigt werden, wie sich die Signale optimal auswerten lassen.
Verwendet werden wieder Sensoren aus dem 35-Sensor-Kit, das bei Elektor erhältlich ist [1]. Alle Softwarebeispiele kann man wie immer von der Webseite zu diesem Artikel [5] herunterladen.

Der Tracking-Sensor

Jeder kennt kleine Roboter auf zwei Rädern, die einer weißen Linie folgen. Dabei wertet man die Reflexionsfähigkeit des Untergrunds aus. Eine IR-Sendediode beleuchtet den Boden, und ein IR-Fototransistor empfängt das reflektierte Licht. Im Prinzip handelt es sich also um eine Reflexlichtschranke. Um die optimale Empfindlichkeit für unterschiedliche Oberflächen und Abstände einzustellen, verwendet der

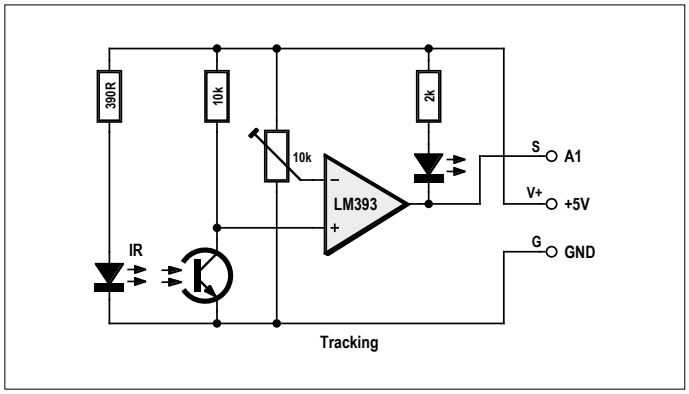


Bild 1. Die einstellbare Reflexlichtschranke.

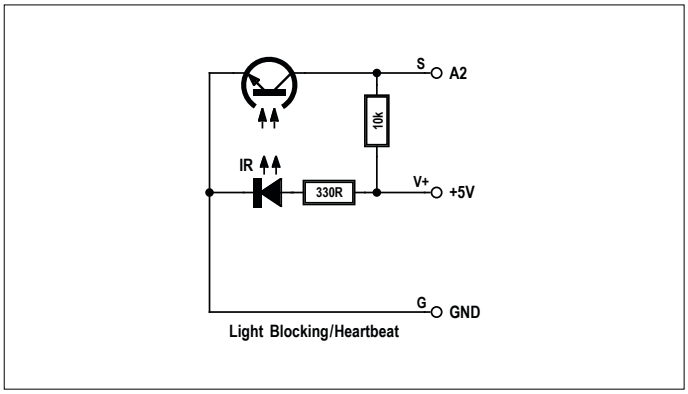


Bild 2. Die Gabellichtschranke.

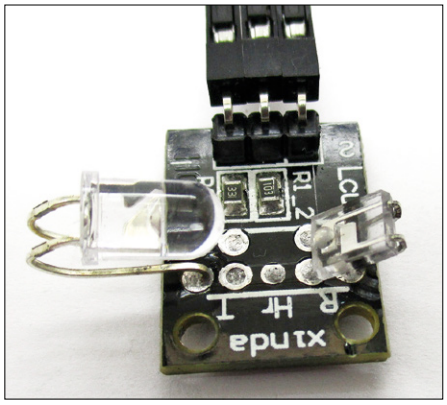


Bild 3. Der vorbereitete Puls-Sensor.

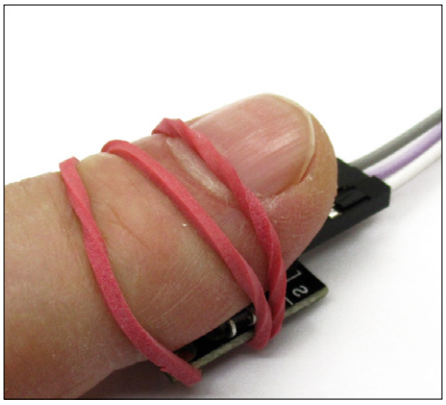


Bild 4. Der Puls-Sensor im Einsatz.

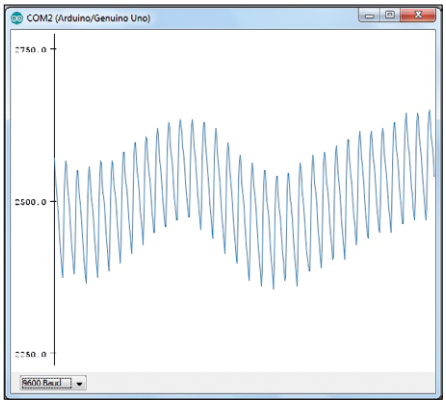


Bild 5. Die Spannung am Pulssensor.

Tracking-Sensor ein Poti und einen Komparator (**Bild 1**). Dank der LED am Ausgang des Komparators kann man die Funktion leicht testen. Als reflektierende Oberfläche reicht bereits ein Finger, den man in einem Zentimeter Abstand vor den Sensor hält. Mit dem Poti stellt man dann den optimalen Umschaltpunkt ein.
Am Ausgang erscheint ein digitales Signal, das in gewohnter Weise ausgewertet werden kann. Man kann die in Folge 2 der Serie [3] vorgestellte Software einsetzen oder auch einen Aktor direkt anschließen. Was am Ende damit gebaut wird, bleibt Ihrer Phantasie überlassen. Es muss ja nicht unbedingt ein Roboter sein. Vielleicht eine Lampe oder ein Ventilator, den man durch Annäherung mit der Hand schaltet, oder was sonst gerade gebraucht wird.

Die Gabellichtschranke

In vielen Maschinen und Geräten (vom Scanner bis zum 3D-Drucker) findet man Endschalter zur Erkennung einer Position eines beweglichen Teils. Mechanische Kontakte oder Mikroschalter haben oft Probleme mit der Langzeitstabilität und müssen nach längerem Einsatz erneuert werden. Mit einer Gabellichtschranke ist man besser bedient, denn die verwendete IR-Diode und der Fototransistor zeigen zumindest bei moderatem LED-Strom keine Alterung.
Eine Sendediode und der Fototransistor (**Bild 2**) stehen sich in geringem Abstand gegenüber. Das gabelförmige Gehäuse enthält einen Schlitz, in den man einen lichtundurchlässigen Gegenstand halten kann. Die Schaltposition wird dabei gut reproduzierbar mit einer Genauigkeit von unter einem Millimeter erfasst. Ähnliche Gabellichtschranken werden auch in Computer-Mäusen eingesetzt und beweisen dort täglich ihre Zuverlässigkeit.
Das Ausgangssignal am Kollektor des Fototransistors kann wahlweise analog oder digital abgefragt werden. Was genau mit dem Signal bewirkt werden soll, hängt von der Aufgabenstellung und der Software ab. Man kann eine Position mit einer Genauigkeit von weniger als 1 mm anfahren - vielleicht genau das, was man für einen 3D-Drucker braucht. In weniger kritischen Fallen reicht eine digitale Abfrage: Licht oder kein Licht.

Der Pulssensor

Pulsmessgeräte verwenden oft ein optisches Messverfahren. Licht einer passenden Wellenlänge wird durch einen Finger oder ein Ohrläppchen gesendet und dann mit einem Fototransistor empfangen. Das Licht wird teilweise im Blut des Körpers absorbiert. Der Herzschlag ändert periodisch die momentane Durchblutung und moduliert daher in geringem Maße den Lichtstrom. Die Signale muss man natürlich analog auswerten, denn sehr große Änderungen sind nicht zu erwarten.
Der Heartbeat-Sensor ist aufgebaut wie eine Gabellichtschranke (vgl. Bild 2), aber mit größerem Abstand zwischen Sender und Empfänger. Sowohl die IR-Sendediode als auch der Fototransistor haben lange Anschlussdrähte, sodass man die optimale Position einstellen kann. Die Platine ist identisch mit der Platine für die Gabelleichtschranke, wobei allerdings der Anschlussstecker und die Bauteile von der anderen Seite bestückt wurden. Zum Betrieb muss man die IR-Diode und den Fototransistor so biegen, dass ein Finger zwischen beiden eingeklemmt werden kann. Wichtig ist aber auch, dass der Finger die Platine an keiner Stelle leitend berührt, weil damit zusätzliche Brummstörungen eingekoppelt werden könnten. Bei uns wurden die Sensoren wie in **Bild 3** gebogen. Die IR-LED leuchtet dann fast direkt auf den Fototransistor. Die Ausgangsspannung ist nun nahe null. Legt man aber den Finger auf, wird das Licht abgeschattet. Der Fototransistor leitet weniger, sodass die Spannung ansteigt. Für einen längeren Versuch ist es sinnvoll, den Sensor mit einem Gummiband am Finger zu halten (**Bild 4**). Wichtig ist nämlich, dass der Finger entspannt ist und kein wechselnder Druck das Signal verfälscht.
Für einen ersten Test mit der Arduino-IDE reicht das Programm VoltageAD2 aus der Folge 1 [2] der Serie. Die Spannung am Fototransistor kann nun mit dem seriellen Plotter dargestellt werden (**Bild 5**). Man sieht deutlich die Lichtmodulation durch den Puls. Überlagert sind aber langsamere Schwankungen, die durch unvermeidliche Bewegungen verursacht werden. Daraus ergibt sich die Herausforderung der Auswertung der Signale.
Zur besseren Verarbeitung wird zunächst ein Tiefpassfilter verwendet (**Listing 1**). Es bildet einfach einen gleitenden Mittelwert der Messwerte. Dazu braucht man nicht mehr als zwei Zeilen. Jeder aktuelle Messwert wird zu einer Mittelwert-Summe `mean` hinzuaddiert, nachdem diese zunächst um 1/20 verkleinert wurde. Jeder einzelne Messwert hat damit nur einen Einfluss von 5 % auf den Mittelwert. Die Zeitkonstante des Tiefpassfilters ist damit die Abtastperiode mal 20, also in diesem Fall 0,4 s. Die Abtastrate wurde nämlich auf 50 Hz festgelegt, um eventuelle Störsignale aus dem Lichtnetz zu dämpfen. Die Grenzfrequenz des Filters liegt bei f = 1 / (2 Pi T), also bei etwa 0,398 Hz. Zur Veranschaulichung werden immer abwechselnd die ungefilterten und die gefilterten Messwerte ausgegeben.

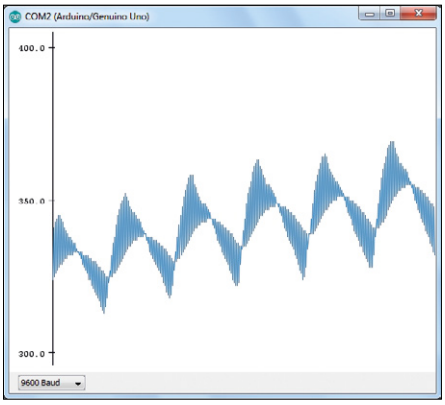


Bild 6. Original und Mittelwert.

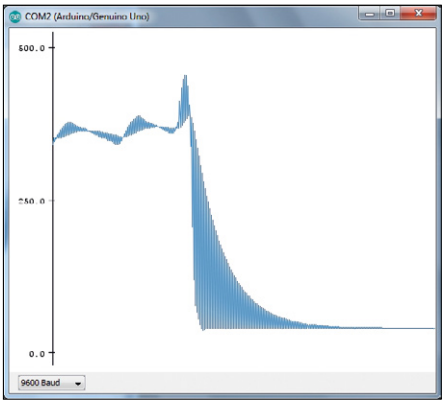


Bild 7. Die Impulsantwort.

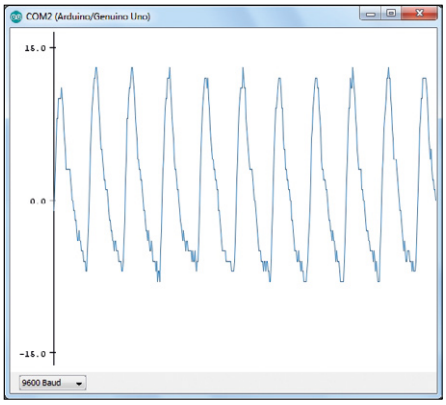


Bild 8. Gefilterte Puls-Signale.

Im seriellen Plotter sieht man nun, wie das Filter arbeitet. Der gleitende Mittelwert folgt dem Signal mit einer gewissen Trägheit (**Bild 6**). Wenn man den Finger schnell aus dem Sensor nimmt, kann man die Impulsantwort des Tiefpassfilters sehen (**Bild 7**).
Nun muss man nur noch die Differenz (Eingangssignal – Mittelwert) bilden (**Listing 2**). Damit realisiert man ein Hochpassfilter und hat dann ein sauberes Puls-Signal, das von allen langsamen Schwankungen befreit ist. Damit kann man nun weiter arbeiten und die Pulsfrequenz auswerten.

Außerdem erkennt man nun weitere Details. Das Signal steigt jeweils steil an und fällt weniger steil wieder ab (**Bild 8**). Man sieht, dass das Herz mit jedem Pulsschlag bei hohem Druck über die Arterien Blut in den Finger pumpt, von wo es dann langsamer über die Venen wieder abfließt. Auch eventuelle Unregel-

Listing 1. Mittelwertbildung.

```
//VoltageAD2 0...1023 at AD2 filter
int sensorPin = 2;
int value;
int mean;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
}
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  Serial.println(value);
  Serial.println(mean / 20);
  delay(19);
}
```

Listing 2. Hochpassfilter.

```
//Filter2AD2 0...1023 at AD2 filter
int sensorPin = 2;
int value;
int mean;

void setup() {
  Serial.begin(9600);
}
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  value = value - mean / 20;
  Serial.println(value);
  delay(19);
}
```

Listing 3. Der Arduino-Pulsmesser.

```
void loop() {
  value = analogRead(sensorPin);
  mean = mean - mean / 20;
  mean = mean + value;
  value = value - mean / 20;
  if ((old < 0) & (value > 0)) {
    time2 = millis();
    pulseTime = time2-time1;
    time1 = time2;
    //Serial.println(pulseTime);
    n = n + 1;
    pulseFreq = 60000 / pulseTime;
    if (pulseFreq > 45) {
      Serial.println(pulseFreq);
      lcd.setCursor(0, 0);
      lcd.print(pulseTime);
      lcd.print(" ms  ");
      lcd.setCursor(0, 1);
      lcd.print(pulseFreq);
      lcd.print(" /min  ");
    }
  }
  old = value;
  delay(20);
}
```

mäßigkeiten im Puls-Rhythmus könnte man leicht erkennen. Um nun einen vollständigen Pulsmesser daraus zu machen, müssen die Abstände der positiven Flanken gemessen werden. Der Arduino kennt eine einfache Zeitmessung mit der Funktion `millis()`. Es wird die Zeit seit dem letzten Reset in Millisekunden zurückgegeben. Damit kann man den Abstand zwischen zwei Pulsschlägen messen. Das Programm (**Listing 3**) zeigt die Pulszeit in ms und die Pulsfrequenz in Schlägen pro Minute auf dem LCD des Elektor Extension-Shields [6] an. Die Frequenz wird für jeden einzelnen Pulsschlag aus der Impulsdauer berechnet, sodass man nicht erst lange warten muss. Es hat sich allerdings gezeigt, dass schon bei leichten Bewegungen Impulse ausfallen, was dann zu zwei- oder dreifachen Pulszeiten führt. Deshalb werden Messwerte nur dann ausgegeben, wenn die Pulsfrequenz größer als 45 Schläge pro Minute ist.

Pulsmessung mit Bascom

In Bascom muss man sich selbst um die Zeitmessung kümmern, denn da gibt es eine permanente Zeitmessung im Hintergrund nur, wenn man sie selbst einrichtet. Dazu eignet sich ein Timer-Interrupt. Hier wird der Timer 0 für die Zeitmessung verwendet. Die Interruptroutine wird alle 10 ms aufgerufen, sodass man die Zeit mit dieser Auflösung erfasst.
Wenn schon Interrupt, dann gleich richtig. In diesem Fall bedeutet das, dass auch die Messungen der Spannung im Interrupt erledigt werden. Das hat den Vorteil, dass sie sehr exakt im 10-ms-Takt stattfinden, sodass eventuelle 50-Hz-Störsignale herausfallen.
10 ms sind eine lange Zeit für einen ATmega, und deshalb kann auch noch die ganze Auswertung im Interrupt durchgeführt werden (**Listing 4**). Das betrifft wie bei der Arduino-Vorlage die gleitende Mittelwertbildung, die Auswertung der Pulsflanken, die Eliminierung von Messfehlern durch ausgefallene Impulse und die Ausgabe im Terminal und auf dem LCD. Am Ende haben wir einen Bascom-Pulsmesser mit ganz ähnlichen Eigenschaften wie im Arduino-Projekt.

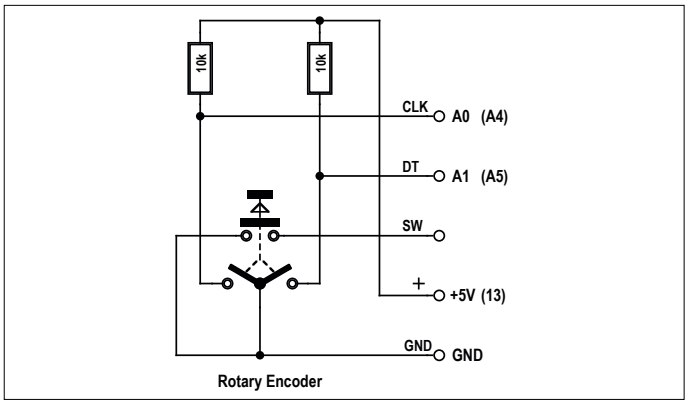


Bild 9. Der Drehencoder.

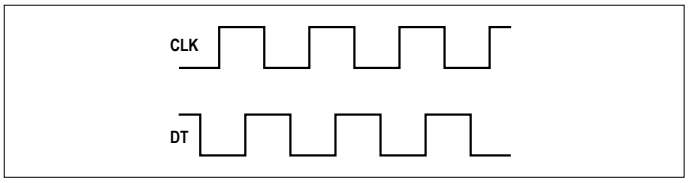


Bild 10. Schaltsignale bei Drehung.

Der Drehencoder

Wer einmal eine Computer-Maus der alten Bauart mit Maus-Kugel geschlachtet hat, ist dort auf Lichtschranken gestoßen, mit denen die Rotation gemessen wurde. Der entscheidende Punkt ist, dass man nicht einfach nur Impulse zählt, sondern mit zwei Lichtschranken auch die Richtung der Drehung unterscheiden kann. Beide Lichtschranken wurden dazu so angeordnet, dass sie bei einer Drehung 90-Grad-versetzte Rechteckimpulse liefern. Genau solche Signale liefert auch der Dreh-Encoder aus dem Sensor-Kit (**Bild 9**). Statt Lichtschranken verwendet der Encoder

Listing 4. Der Bascom-Pulsmesser.

```
'---------------------------------
'Pulse.BAS  ADC2, LCD
'---------------------------------

...

Config Timer0 = Timer , Prescale = 1024
On Ovf0 Tim0_isr
Enable Timer0
Enable Interrupts

...

Do
Loop

Tim0_isr:
  Timer0 = 100     '10ms
  D = Getadc(2)
  Du = U / 40
  U = U - Du
  U = U + D
  Ticks = Ticks + 1
  U1 = D - Du
  If U1 > 0 And U2 < 0 Then
  T = Ticks * 10
     F = 60000 / T
     Ticks = 0
     If F > 45 Then
        Print T
        Print F
        Locate 1 , 1
        Lcd T
        Lcd " ms "
        Locate 2 , 1
        Lcd F
        Lcd " /min "
     End If
  End If
  U2 = U1
Return
```

allerdings mechanische Kontakte. Und er hat eine deutlich spürbare Rasterung, sodass man beim Drehen die einzelnen Stufen spürt und mitzählen kann. Auf eine Umdrehung kommen übrigens 20 Stufen. Das mag wenig erscheinen, aber weil die Achse keinen Endanschlag hat, kann man damit auch Hunderte oder Tausende von Stufen anwählen, wie das bei der genauen Einstellung einer Frequenz oder einer Spannung sinnvoll ist. Zusätzlich gibt es noch einen Drucktaster, der aber hier nicht verwendet wird.

Obwohl die beiden Kontakte eigentlich gleichwertig sind, wird einer als Takt-Pin CLK gekennzeichnet, der andere als Daten-Pin DT. Das **Bild 10** zeigt die Ausgangssignale bei einer Drehung. Die Auswertung ist ganz einfach: Man wartet auf eine fallende Flanke an CLK und schaut dann wie DT gerade steht, um entsprechend einen Zählerstand entweder zu erhöhen oder zu verkleinern.

Das Beispielprogramm (**Listing 5**) verwendet gleich zwei Encoder, mit denen man zwei analoge Ausgangssignale PWM1 und PWM2 an den Arduino-Ausgängen 9 und 10 einstellen kann. Das könnte dann mit einem zusätzlichen Tiefpassfilter eine zweifache einstellbare Spannungsquelle für allgemeine Messzwecke werden. Man kann damit auch die Mischfarbe der Rot/Grün-LED aus dem Sensor-Kit feinfühlig und reproduzierbar einstellen. Die eingestellten Werte werden zusätzlich seriell ausgegeben und auf dem LCD angezeigt.

Der Dreh-Encoder besitzt eigene Pullup-Widerstände. Trotzdem werden auch beim Arduino die internen Pullups eingeschaltet. So kann man nämlich mit demselben Programm wahlweise einen oder zwei Encoder verwenden. Der erste liegt an den Eingängen A0 und A1. Rein zufällig liegen hier auch die Taster S1 und S2 auf dem Extension-Shield. Weil alle Kontakte an GND liegen, gibt es damit keinen Konflikt. Aber man kann notfalls den ersten Ausgabewert d1 auch mit den Tasten verstellen. S1 liefert dann das Taktsignal und S2 die Drehrichtung. Den zweiten Encoder haben wir an A4 und A5 angeschlossen. Die entsprechenden Portpins sind beim Arduino Uno zweimal

Listing 5. Encoder-Auswertung in Arduino-C.

```
//Encoder  A0/A1 PMW1, A4/A5 PWM2

...

int clk1 = A0;
int dt1 = A1;
int clk2 = A4;
int dt2 = A5;
int pwm1 = 9;
int pwm2 = 10;

...

void loop() {
  new1 =  digitalRead(clk1);
  if((new1==0) & (old1==1)){
    if (digitalRead(dt1)==0)  d1++; else  d1--;
    Serial.println (d1);
    if (d1 > 250) d1 = 250;
    if (d1 < 0) d1 = 0;
    analogWrite (pwm1, d1);
    lcd.setCursor(0, 0);
    lcd.print(d1 * 20);
    lcd.print(" mV  ");
  }
  old1=new1;

  new2 =  digitalRead(clk2);
  if((new2==0) & (old2==1)){
    if (digitalRead(dt2)==0)  d2++; else  d2--;
    Serial.println (d2);
    if (d2 > 250) d2 = 250;
    if (d2 < 0) d2 = 0;
    analogWrite (pwm2, d2);
    lcd.setCursor(0, 1);
    lcd.print(d2 * 20);
    lcd.print(" mV  ");
  }
  old2=new2;
}
```

Listing 6. Encoder-Auswertung in Bascom.

```
'Encoder.BAS    C0/C1 PMW1a, C4/C5 PWM1b

...

Do
  New1 = Pinc.0
  If New1 = 0 And Old1 = 1 Then
    If Pinc.1 = 0 Then D1 = D1 + 1 Else D1 = D1 -1
    If D1 > 1023 Then D1 = 1023
    If D1 < 0 Then D1 = 0
    Pwm1a = D1
    Print D1
    Locate 1 , 1
    Lcd D1
  End If
  Old1 = New1

  New2 = Pinc.4
  If New2 = 0 And Old2 = 1 Then
    If Pinc.5 = 0 Then D2 = D2 + 1 Else D2 = D2 -1
    If D2 > 1023 Then D2 = 1023
    If D2 < 0 Then D2 = 0
    Pwm1b = D2
    Print D2
    Locate 2 , 1
    Lcd D2
  End If
  Old2 = New2
Loop
```

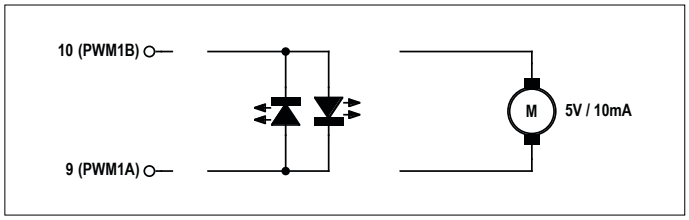


Bild 11. Brückenschaltungen.

herausgeführt, nämlich auch diagonal gegenüber als SDA und SCL der I2C-Schnittstelle. Ganz in der Nähe findet sich auch ein GND-Pin. Was aber eigentlich fehlt, ist ein zusätzlicher 5-V-Anschluss, den man ja für die Pullups auf dem Encoder braucht. Aber kein Problem, da wird einfach der Pin 13 hochgeschaltet, an dem auch die Arduino-LED liegt. Das geht immer: Wenn mal GND oder VCC fehlt, programmiert man sich einen Portpin zu diesem Zweck.

Encoder in Bascom

Die Bascom-Version des Programms ist fast gleich aufgebaut. Man kann zwischen den Sprachen wechseln, indem man gegenseitig den Quellcode übernimmt, die Variablennamen beibehält und nur die spezifischen Syntax-Eigenheiten überarbeitet. Im Endergebnis gibt es nur einen echten Unterschied: In Bascom verwendet man die beiden PWM-Ausgänge von Timer1 üblicherweise mit einer Auflösung von 10 bit, sodass man nun 1023 Schritte und eine Schrittweite von rund 5 mV hat.
Zwei genau einstellbare PWM-Ausgänge, das ruft geradezu nach einem Experiment zum Thema Brücken-Endstufe, zumal die beiden PWM-Signale vom selben Timer stammen und exakt zueinander passende Impulse mit einer zeitgleich steigenden Flanke haben. Man könnte eine grüne und eine rote LED antiparallel zwischen beide Ausgänge legen (**Bild 11**). Stellt man eine mittlere Spannung wie z.B. 2500 mV auf beiden Kanälen ein, bleiben beide LEDs aus. Bringt man aber einen Kanal aus dem Gleichgewicht, beginnt eine der LEDs zu leuchten.
Genauso könnte man einen extrem sparsamen DC-Motor (Typ Kassettenrecorder, Anlaufstrom 10 mA) anschließen und dann Richtung und Drehzahl steuern. Ja stimmt, sowas tut man eigentlich nicht ohne einen richtigen Motortreiber. Aber es funktioniert tatsächlich. Eine Gefahr durch Induktionsspannungen besteht in diesem Fall nicht, weil die Ausgänge sich immer in einem niederohmigen Zustand befinden. Außer vielleicht, wenn man im laufenden Betrieb den Arduino resettet, denn dann werden die Anschlüsse hochohmig. Also gut, besser nur mal kurz testen und den Motor wieder entfernen, solange das Programm noch läuft. ◀

(160302)

Weblinks

[1] www.elektor.de/arduino-sensor-kit
[2] www.elektormagazine.de/160152
[3] www.elektormagazine.de/160173
[4] www.elektormagazine.de/160210
[5] www.elektormagazine.de/160302
[6] www.elektormagazine.de/140009

Anzeige

HOCH-LEISTUNGS-8-KANAL-OSZILLOSKOP

pico Technology

12-Bit-Auflösung

20 MHz Bandbreite

8 Kanäle

256 MS Pufferspeicher

Das PicoScope 4824 ist ein preiswertes, tragbares Gerät für mehrere Eingangsanwendungen. Mit 8 hochauflösenden Analog-Kanälen können Sie problemlos Audio-, Ultraschall-, Stromzufuhr- und Timing-Signale komplexer Systeme analysieren.

Das PicoScope 4824 verfügt über die Leistung und Funktionalität, um hochpräzise Ergebnisse zu liefern. Zudem zeichnet es sich durch einen großzügigen Speicher aus, um mehrere seriellen Bus-Signaltypen wie UART, I²C, SPI, CAN und LIN, sowie Steuer- und Treibersignale zu analysieren.

Weitere Merkmale sind:
- Leistungsstarker Generator für anwenderdefinierte Wellenformen
- Erweiterte digitale Trigger
- Serielle Bus-Entschlüsselung
- SuperSpeed USB 3,1 Gen 1-Schnittstelle
- Windows-, Mac- and Linux-Software

Neues Software-Update
Fordern Sie die neuesten Softwareversionen an, unter
www.picotech.com/library/picoscope/picoscope-release-6.12.5

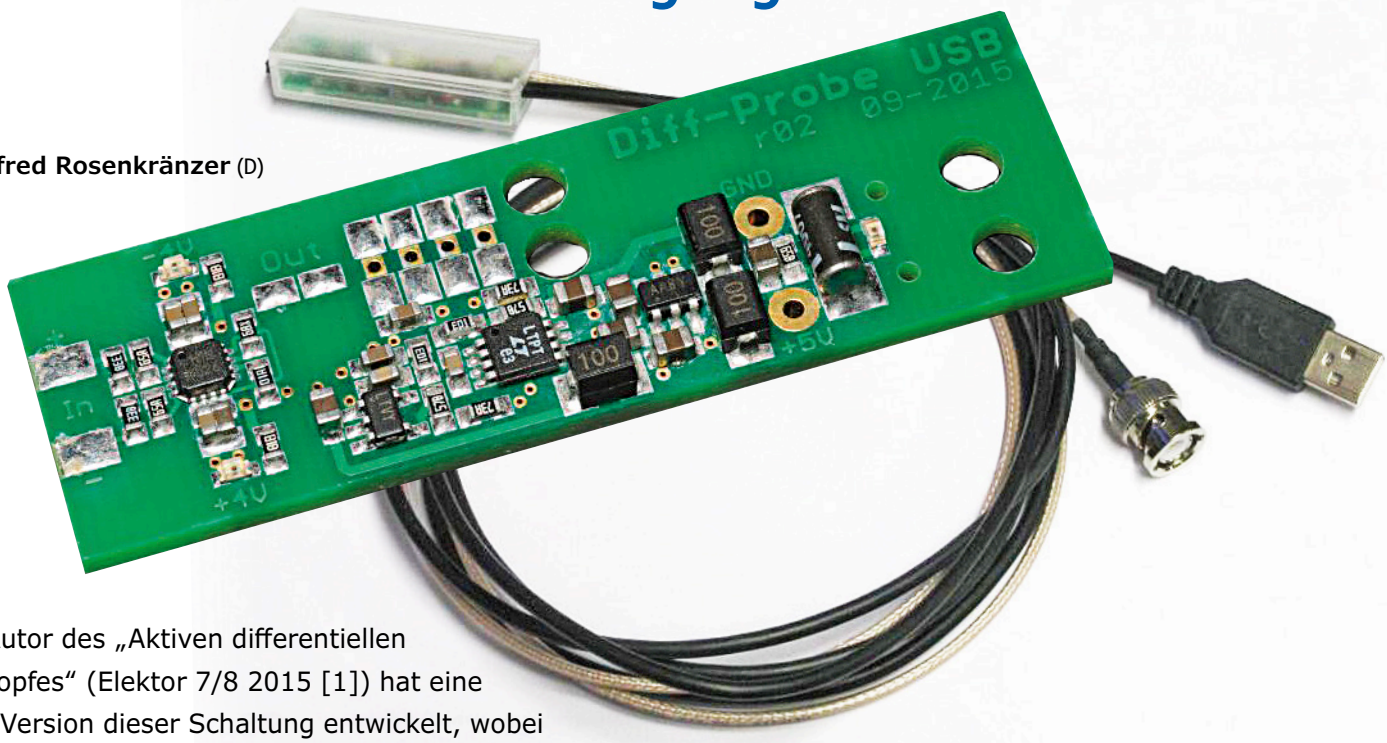
Für weitere Informationen, besuchen Sie uns bitte unter
www.picotech.com/PS504

Die Preise gelten zum Zeitpunkt der Drucklegung, MwSt. nicht inbegriffen. Bitte erkundigen Sie sich vor der Bestellung bei Pico Technology nach den aktuellen Preisen. E-Mail: sales@picotech.com Fehler und Auslassungen vorbehalten.

Aktiver differentieller Tastkopf V2

Jetzt mit USB-Stromversorgung

Von **Alfred Rosenkränzer** (D)



Der Autor des „Aktiven differentiellen Tastkopfes“ (Elektor 7/8 2015 [1]) hat eine neue Version dieser Schaltung entwickelt, wobei die Stromversorgung nun über die 5 V eines USB-Anschlusses erfolgt.

In **Bild 1** ist das Blockschaltbild der neuen Stromversorgung zu sehen. Nachdem die 5-V-Spannung am USB-Stecker gefiltert wurde, erzeugt eine „Charge Pump“ eine ungeregelte negative Spannung in (etwa) der gleichen Größe wie die Eingangsspannung. Nach nochmaliger Filterung regelt ein negativer Low-Drop Regler sie auf -4 V herunter. Ein positiver Low-Drop-Regler erzeugt +4 V direkt aus der gefilterten USB-Spannung.

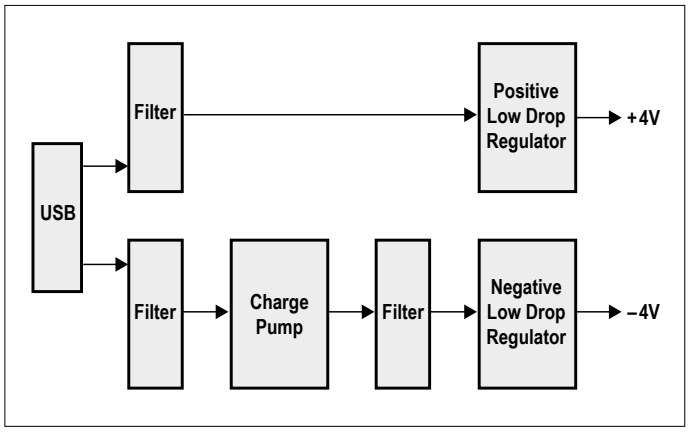


Bild 1. Blockschaltbild der neuen USB-Stromversorgung.

Bild 2 zeigt die komplette Schaltung. IC1, ein LT1962EMS8, liefert die stabilisierte positive Speisespannung. IC2, ein LT1964ES5, erfüllt die gleiche Funktion für die negative Spannung. Als Charge Pump dient ein MAX1697 (IC3).

Das Vorhandensein der USB-Spannung als auch der beiden geregelten Spannungen wird durch LEDs (D2, D3 und D4) angezeigt. Für die Filterung der „rohen“ USB-Spannung sorgen die Spulen L1 und L2.

Die Schaltung des Differenzverstärkers (IC4) ist im Wesentlichen gleich geblieben. Lediglich die beiden in Reihe geschalteten Eingangswiderstände beider Eingänge wurden jeweils in einem Bauteil zusammengefasst, so dass der Opamp auf der Platine nach vorne gerückt werden konnte, um Platz für die Spannungsversorgung zu schaffen.

Gegenüber der in Elektor 7-8/2015 veröffentlichten Schaltung ergeben sich einige Änderungen. Da keine isolierenden Spannungswandler verwendet werden, ist die Masse des USB-Anschlusses mit der Signalmasse verbunden.

Trotz aufwendiger Filterung ist die Taktfrequenz der Charge Pump von etwa 200 kHz mit einem Spektrumanalyser bei

< −80 dBm zu sehen. Auf einem Oszilloskop ist sie jedoch nicht zu entdecken.
Und schließlich erhöht sich auch der Preis durch die teuren Regler. Die Schaltung passt weiterhin in das bereits verwendete USB-Gehäuse, auch das BNC-Kabel ist gleich geblieben. ⏮

(150801)

Für interessierte Leser bietet der Autor fertig aufgebaute Tastköpfe an.
Info: alfred_rosenkraenzer@gmx.de.

Weblink

[1] www.elektormagazine.de/130538

Spezifikationen:

Dämpfung:	10:1 bei differentiellem Signal und 50 Ω Abschluss
Differentieller Eingangswiderstand:	5100 Ω, ±1%
Single-Ended-Eingangswiderstand:	2550 Ω, ±1%
Eingang Common Mode Range:	±12 V
Ausgangswiderstand:	50 Ω, ±1%
Bandbreite:	ca. 1,9 GHz (-3 dB)
Rise/Fall-Time:	300 ps
Stromversorgung:	USB 5 V, ca. 70 mA

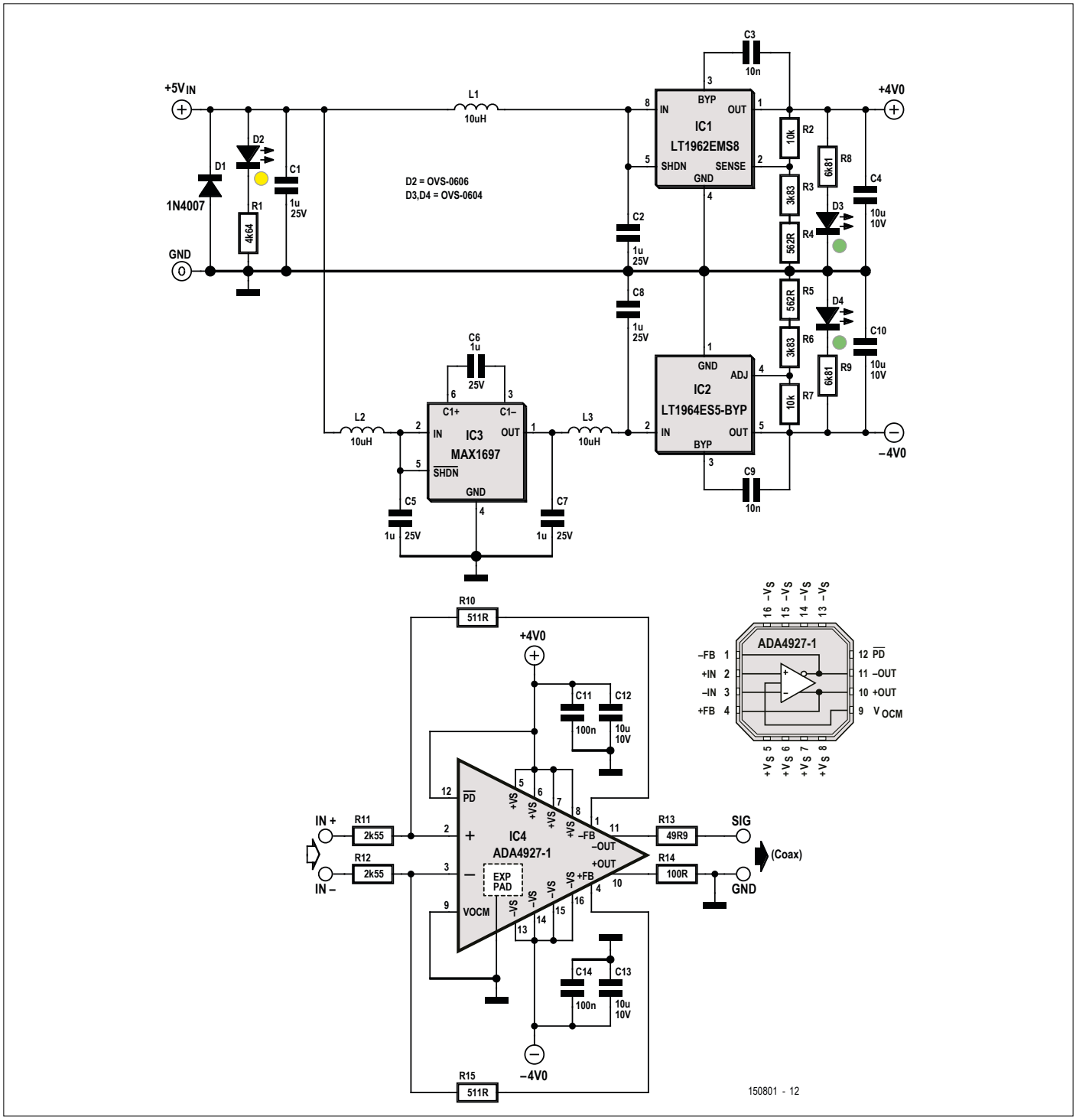


Bild 2. Die komplette Schaltung. Der Teil rund um den Differenzverstärker (IC4) ist im Wesentlichen gleich geblieben.

RFID-Starterkit für Arduino Uno

Temperaturmessung und Zugangskontrolle

Mit dem RFID-Starterkit bietet Elektor eine Grundausstattung für interessante Elektronik-Experimente mit dem Arduino Uno an. Wir zeigen anhand zweier Beispiele, wie leicht sich eine Applikation verwirklichen lässt.

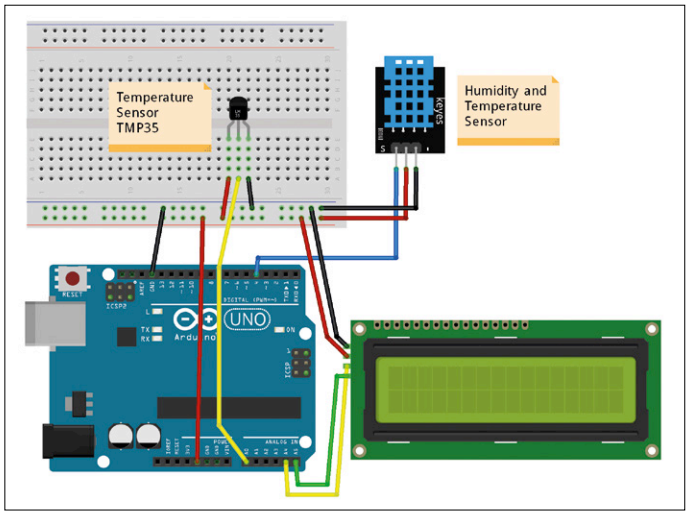


Bild 1. Klimastation mit zwei Sensoren und Display.

Der Arduino und seine Nachbauten, die sogenannten Klone, sind seit vielen Jahren eine feste Größe in der Gemeinde der Elektroniker und der Maker. Wo früher reine Mikrocontroller eingesetzt wurden, kommt heute immer häufiger der Arduino zum Zug. Der Einstieg in die Welt der Controllertechnik ist damit zum Kinderspiel geworden. Anstelle einer komplexen Programmierumgebung kann man jetzt mit einem intuitiven Entwicklungstool arbeiten, das auch Anfängern keine Probleme bereitet.
Etwas anders sieht es mit den Hardwareerweiterungen aus. Hier ist der Anwender immer noch weitgehend auf sich alleine gestellt. Zwar existieren zahlreiche Shields für eine Fülle von speziellen Erweiterungen, allerdings sind diese oft auf wenige Anwendungsgebiete beschränkt. Wenn man wirklich innovative Projekte umsetzen will, muss man sich mit den elektronischen Bauelementen befassen, was aber viele Einsteiger vor größere Probleme stellt.

Abhilfe bieten hier „Kits“, die alle notwendigen Komponenten enthalten, die zur Durchführung verschiedener Eigenbauprojekte notwendig sind. Solche Kits begannen in den 70er und 80er Jahren als sehr beliebte Elektronikbaukästen. Wo in früheren Jahren Sirenen und Mittelwellenradios gebaut wurden, entstehen nun die verschiedensten Mikrocontroller-Projekte.
Eine besonders interessante Variante eines neuen Baukastens ist das „RFID Starter Kit für den Arduino Uno“. Es enthält über 30 Komponenten, Bauelemente und Module aus allen Bereichen der modernen Elektronik. Die Bezeichnung des Kits ist sogar etwas irreführend, denn der Bauelementesatz enthält zwar auch eine RFID-Empfängerplatine und zwei dazugehörende Transponderchips in Form einer Karte und eines Schlüsselanhängers, aber darüber hinaus auch eine Fülle weiterer Bauelemente.
Neben dem Arduino Uno selbst findet sich in der praktischen Kunststoffbox unter anderem:

- ein Feuchtigkeitssensor,
- eine Multicolor-LED,
- eine große LED-Matrix mit 64 integrierten Leuchtpunkten,
- eine vierstellige Siebensegmentanzeige,
- eine Infrarot-Fernbedienung sowie ein dazu passender Empfänger,
- ein komplettes LC-Display-Modul mit I2C-Anschluss.

Detaillierte Informationen zum Inhalt des Kits sind unter [1] zu finden. Die beiden hier beschriebenen Beispiele stellen natürlich nur einen winzigen Ausschnitt aus den vielfältigen Mög-

lichkeiten des Kits dar, mit dem sich eine nahezu unbegrenzte Anzahl von Experimenten und Anwendungen aufbauen lässt.

Universelle Klimastation mit LC-Display

Im ersten Anwendungsbeispiel werden mehrere Sensoren eingesetzt, um verschiedene Umweltdaten zu erfassen. Dabei kommt der Temperatur/Feuchtesensor DHT11 zum Einsatz, außerdem der Einzeltemperatursensor LM35DZ, so dass man beispielsweise die Temperatur und Luftfeuchtigkeit in einem Raum sowie die Außentemperatur simultan überwachen kann. Für die Anzeige der Werte wird das LC-Display eingesetzt. Der vollständige Hardware-Aufbau ist in **Bild 1** dargestellt, der Sketch ist in **Listing 1** wiedergegeben.

Der **Sensor LM35** wird vom Hersteller genau auf eine Steigung von 10 mV/°C kalibriert. Zudem liegt der Nullpunkt der Ausgangsspannung auf genau 0,0 °C. Zusammen mit der Auflösung des verwendeten ADCs von 10 bit (Werte 0...1023) und der Versorgungsspannung des LM35 von 5 V ergibt sich daraus die Formel:

```
temp = (5.0 * analogRead (tempPin) * 100.0) / 1023;
```

Damit wird jedoch nur eine geringe Auflösung erreicht. Der LM35 erzeugt nämlich nur Spannungen von 0 V bis etwas über +1 V. Wird nichts anderes angegeben, dann verwendet der arduino-interne ADC seine Versorgungsspannung von 5 V als Referenz, so dass 80 % des Wertebereiches nicht genutzt werden. Wählt man die interne Referenzspannung von 1,1 V, dann erreicht man nahezu die maximale Auflösung.

Wird eine Referenz von 1,1 V verwendet, ändert sich natürlich auch die Umrechnungsformel. Teilt man die 1,1 V durch 1023, entspricht jeder Schritt in der digitalen Ausgabe etwa 0,001075 V beziehungsweise 1,0752 mV. Bei einer Steigung von 10 mV/°C ergibt sich der Umrechnungsfaktor:

```
float TempCal = 0.1075;
   //1.075 / 10
```

Die Berechnung der Temperatur erfolgt dann über die Programmzeile:

```
tempLM35 = (LM35val * TempCal);
```

Um die interne Referenzspannung von 1,1 V zu aktivieren, erhält der Arduino den Befehl

```
analogReference (INTERNAL);
```

Mit diesem Verfahren ist eine Auflösung von etwa einem Zehntel Grad Celsius erreichbar. Natürlich kann die Referenzspannung eine gewisse Toleranz aufweisen, was sich aber durch Feinjustierung des Wertes `0.1075` ausgleichen lässt. Wurde der exakte Wert erst einmal korrekt bestimmt, kann man über lange Zeit hinweg eine präzise Anzeige erwarten, da die Referenzspannung nur geringen Schwankungen unterliegt.

Der Temperaturbereich beträgt bei diesem Auswerteverfahren 0...110 °C. Da ein Zehntel Grad einer Spannung von nur 1 mV entspricht, sollte man berücksichtigen, dass bereits Thermospannungen an Kontakten und Lötstellen einen gewissen Einfluss haben können. Für ein Zimmerthermometer ist eine Anzeige von ganzen Graden zwar ausreichend, bei präziseren Messungen sollten die oben stehenden Hinweise allerdings nicht unbeachtet bleiben.

Der kombinierte **Temperatur/Feuchtesensor DHT11** wird über den I2C-Bus ausgelesen. Eine fertige Bibliothek macht das Erfassen der Messwerte sehr einfach. Auch das Display ist am I2C-Bus angebunden und auch hierfür gibt es eine passende Bibliothek. Die Ausgabe von Texten oder Messwerten kann so einfach über die Funktion

Listing 1. Umweltgrößen ermitteln mit DHT11 und LM35.

```
// DHT11_LM35_w_LCD_display.ino

#include <dht.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2); // LCD address 0x27
dht DHT;
const int DHT11_PIN= 4;
const int A0 = 0;
float tempLM35 = 0;
long LM35val = 0;
float TempCal = 0.1075;

void setup()
{ lcd.begin();
  lcd.backlight();
  analogReference(INTERNAL);
}

void loop()
{ DHT.read11(DHT11_PIN);

  LM35val = analogRead(A0);
  tempLM35 = (LM35val * TempCal);

  lcd.setCursor(0, 0); lcd.print("Ti="); lcd.print(DHT.temperature,0);
  lcd.print(char(223)); lcd.print("C");    // print unit " °C "

  lcd.setCursor(8, 0); lcd.print("Ta="); lcd.print(tempLM35,0);
  lcd.print(char(223)); lcd.print("C");    // print unit " °C"

  lcd.setCursor(0, 1);
  lcd.print("Humidity:  ");
  lcd.print(DHT.humidity,0); lcd.print(" %");
  delay(200);
}
```

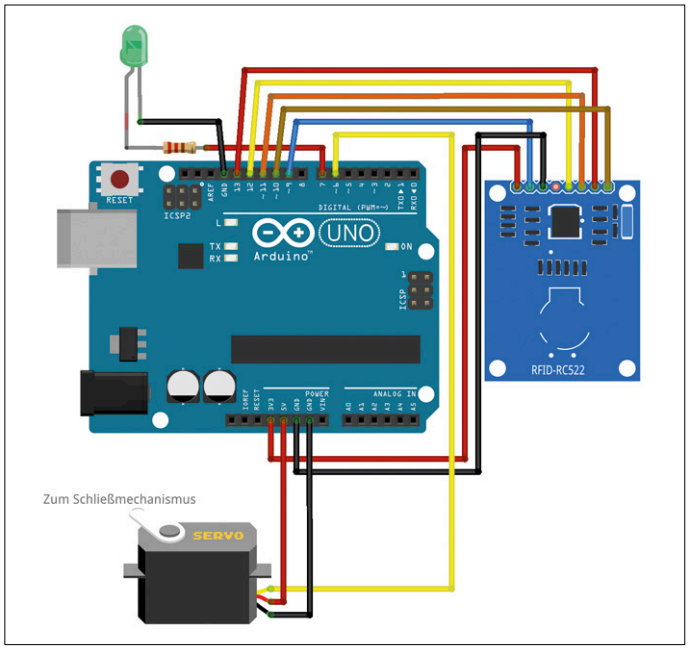


Bild 2. Servo-Steuerung mit RFID-Modul.

```
lcd.print()
```

erfolgen. Eine integrierte Cursorsteuerung ermöglicht eine problemlose Formatierung der Ausgabe. Im erstaunlich kompakten Sketch kommen die Bibliotheken oder „Libraries“ LiquidCrystal_I2C.h [3], DHTLib [4] und Wire.h zum Einsatz. Die Wire-Library wird für die Kommunikation über den I2C-Bus benötigt und gehört (wie auch die Bibliotheken SPI und Servo im folgenden Sketch) zu den Standard-Bibliotheken, die schon in der Arduino-IDE enthalten sind. Die anderen beiden Bibliotheken können kostenlos aus dem Internet geladen werden. Falls sich ein Link einmal als Sackgasse erweisen sollte, finden Sie mit einer Suchmaschine meist schnell Ersatz.
Bei Bedarf kann die Klimastation um weitere Sensoren aus dem RFID-Kit erweitert werden. So kann der Tropfensensor oder aber auch der Fotowiderstand (LDR) zusätzliche Informationen liefern. Mit dem Tropfensensor lässt sich ermitteln, ob es gerade regnet. Der LDR gestattet die Erfassung aktueller Helligkeitswerte, woraus Rückschlüsse auf die Sonnenscheinintensität und -dauer gezogen werden können.

Türschloss mit RFID-Sicherung

Eine weitere sehr interessante Anwendung der Bauteile aus dem Kit ist die Steuerung eines elektronischen Schlosses, das mit einem RFID-Chip geöffnet werden kann. Das beiliegende RFID-Empfängermodul gab dem Kit sogar seinen Namen, obwohl es nur eine von vielen nützlichen und interessanten Komponenten im „RFID Starter Kit“ ist.

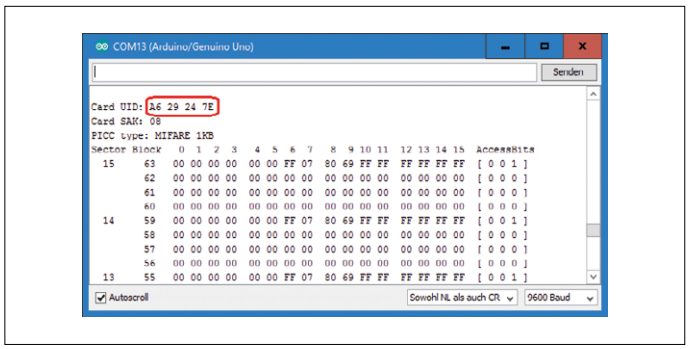


Bild 3. Daten des RFID-Transponders im Serial Monitor.

Die Kommunikation des RFID-Moduls mit dem Arduino erfolgt über das „Serial Peripheral Interface“ oder kurz SPI. Diese Schnittstelle verfügt über drei gemeinsame Leitungen, an denen alle Busteilnehmer angeschlossen sind. Über die Steuerleitung SCK (Serial Clock), wird vom Master zur Synchronisation ein fester Takt ausgegeben. Über MOSI (Master Output, Slave Input) und MISO (Master Input, Slave Output) werden die Daten gesendet und empfangen.
Zu den Takt- und Datenleitungen kommt noch für jeden Slave-Busteilnehmer eine Chip-Select-Leitung. All diese CS-Leitungen (manchmal auch als SS – Slave Select bezeichnet) werden vom Master gesteuert. Über eine Reset-Leitung (RST) kann das Modul zurückgesetzt werden. Dazu kann noch eine Interrupt-Leitung (IRQ) kommen, die hier allerdings nicht verwendet wird.
Das RFID-Modul wird mit dem Arduino wie in **Tabelle 1** gezeigt verbunden.

Softwareseitig sind wieder zwei Bibliotheken erforderlich, nämlich die SPI-Lib aus der Arduino-IDE und die MFRC522-Lib für das RFID-Modul von [5].
Wenn die Hardware wie in **Bild 2** aufgebaut ist, kann der Sketch „RFID-RC522_data“ (**Listing 2**) aus dem Download-Paket [6] auf den Arduino geladen werden. Damit lassen sich bereits Daten aus einem RFID-Transponder lesen, wenn man den Seriellen Monitor der Arduino IDE öffnet. Sobald ein Transponder in die Nähe des RFID-Moduls gehalten wird, erscheinen nun die auf dem Chip gespeicherten Daten (**Bild 3**).
Ein wichtiger Parameter, der vom Sketch verwendet wird, ist die UID (**U**nique **I**dentification **N**umber) des Transponders. Die anderen auf dem Chip gespeicherten Daten sind für unsere Anwendung nicht von Bedeutung.

Eine der wichtigsten Aufgaben für RFID-Module sind Zugangskontrollsysteme. Jede Person, die befugt ist, einen bestimmten Raum zu betreten, erhält einen eigenen Tag in Form einer Karte oder eines Schlüsselanhängers. An der betreffenden Tür muss dann nur noch ein RFID-Modul angebracht werden, das den Schließmechanismus nur dann öffnet, wenn ein gültiger RFID-Tag in seine Reichweite kommt. Gegenüber klassischen Schlössern mit passenden Schlüsseln hat dies mehrere Vorteile: Geht ein ID-Tag verloren, kann man den Code einfach aus der Liste der gültigen IDs streichen. Ein Austausch der Schließanlage ist nicht erforderlich. Und man kann auch Personen durch einfaches Sperren des betreffenden Codes ausschließen, die noch im Besitz eines Tags sind und ihn aus irgendwelchen Gründen nicht herausrücken wollen. Die Tags sind zudem wesentlich preisgünstiger als aufwendige Sicherheitsschlüssel.
Um eine bestimmte ID zu erkennen, ist es vorteilhaft, die Hex-Sequenz der UIDs in eine Dezimalzahl umzuwandeln. Dann kann die Gültigkeit eines bestimmten Tags leicht überprüft werden. Zwar wird dadurch das System etwas weniger sicher, da nicht alle Zahlen der vier Hex-Blöcke eindeutig in eine zusammenhängende Zahl umgewandelt werden, aber wenn man Wert auf maximale Sicherheit legt, kann man den Programmcode entsprechend ändern. So wäre es besser, jeden einzelnen Block abzufragen, der Aufwand dazu ist allerdings deutlich höher.

Der Sketch „RFID-RC522_servo_lock.ino“ findet sich ebenfalls im Download-Paket zu diesem Artikel. Er gibt sowohl die UID als auch den zugehörigen Code als Dezimalzahl im Serial Monitor aus. Die Dezimalzahl ist dann in der Zeile

```
long validCode = 938350; // enter valid code
```

als gültiger Code in die Variable „validCode“ einzutragen. Man kann das System als elektronisches Türschloss verwenden, da die Entriegelungssequenz nur aktiv wird, wenn die Karte mit dem richtigen Code in die Nähe des Empfängermoduls gehalten wird.
Zum RFID-Kit gehört auch ein Modellbauservo, den man zur Ansteuerung eines Türriegels verwenden kann. Wie der Servo mit dem Arduino zu verbinden ist, ist ebenfalls in Bild 2 zu sehen. Zusätzlich wurde noch eine grüne LED angeschlossen, die signalisiert, dass die Tür (oder der Schließmechanismus eines Schließfachs) entriegelt wurde. Dazu wurde der Code um die Servo-Bibliothek und den simplen Befehl am Ende des Sketches erweitert.

Zusammenfassung und Ausblick

Moderne Elektronik-Kits enthalten meist keine einzelnen elektronischen Komponenten mehr, sondern fertige Module. Damit ist man zwar nicht mehr ganz so nahe an der Technik dran, kann aber mit wenigen Handgriffen sehr nützliche und praxistaugliche Projekte umsetzen. Dank umfangreicher Bibliotheken können die zugehörigen Sketche schnell und einfach programmiert werden.

Auf diese Art und Weise haben die Experimentierkasten-Klassiker aus den Anfangsjahren der Elektronik würdige Nachfolger gefunden. Sowohl jungen Nachwuchstüftlern und angehenden Ingenieuren als auch den Alten Hasen der Elektronik-Zunft stehen damit alle Möglichkeiten der modernen Elektronik offen.

Natürlich kann im Rahmen dieses kleinen Artikels nur ein winziger Bruchteil der Möglichkeiten des Kits vorgestellt werden. Im Buch „Praxisprojekte mit dem RFID-Starterkit für Arduino“ [2] werden über 30 Projekte mit allen im Kit vorhandenen Modulen vorgestellt und detailliert erläutert. Zudem findet sich dort eine große Anzahl von Anwendungen und Projekten, die ohne große Vorkenntnisse nachgebaut werden können. Die zugehörigen Sketche und Programme werden genau erläutert, so dass man beim Durcharbeiten des Buches sein Fachwissen immer wieder ergänzen und erweitern kann. Dann endlich steht auch der Entwicklung eigener Geräte und Systeme nichts mehr im Wege!

(160322)

Tabelle 1. Verbindung von RFID-Modul und Arduino.

RC522-Pin	Pin Arduino Uno
VCC (3V3)	3,3 V
GND	GND
RST	9
SDA (SS)	10
MOSI	11
MISO	12
SCK	13

Weblinks

[1] RFID-Starterkit: www.elektor.de/rfid-starter-kit-for-arduino-uno

[2] Projekte mit dem RFID-Starterkit für Arduino, Über 30 Praxisprojekte, Elektor, 2016: www.elektor.de/projekte-mit-dem-rfid-starterkit-fuer-arduino

[3] LiquidCrystal-I2C-Bibliothek: https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

[4] Bibliothek für den Temperatur/Feuchte-Sensor DHT11: http://arduino.cc/playground/Main/DHTLib

[5] Lib für das RFID-Modul: https://github.com/miguelbalboa/rfid

[6] Projektseite mit Downloadpaket: www.elektormagazine.de/160322

Listing 2. Anzeige der RFID-UID im Serial Monitor.

```
// RFID-RC522_data.ino

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9
#define SS_PIN          10

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{  Serial.begin(9600);
   SPI.begin();    // Start SPI bus
   mfrc522.PCD_Init();    // Initialise MFC522 Reader
   mfrc522.PCD_DumpVersionToSerial();    // Show RC522 details
   Serial.println("Place RFID TAG in range!");
}

void loop()
{  // Card present?
   if ( ! mfrc522.PICC_IsNewCardPresent())
   { return; }

   // Select card
   if ( ! mfrc522.PICC_ReadCardSerial())
   { return; }

   // Send data to serial interface
   mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Lautsprecherboxen berechnen

Software gratis im Netz

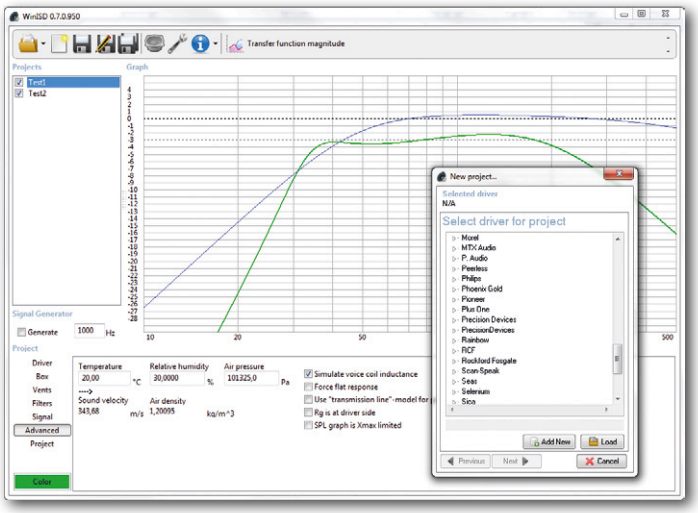
Von **Harry Baggen** (Elektor-Labor)

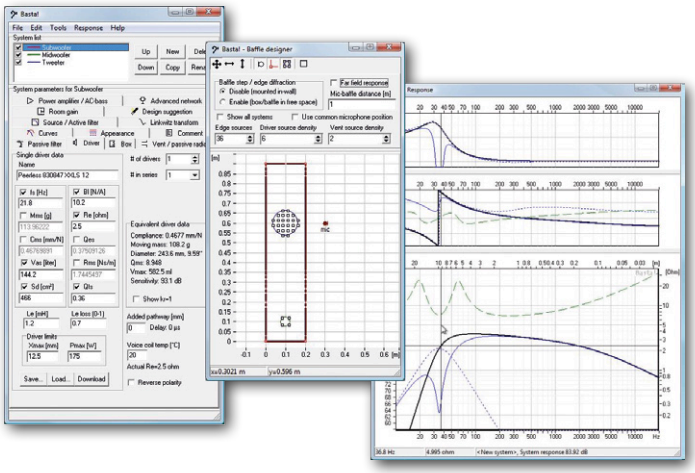
Ein Vorläufer unserer Reihe Web-Recherche startete im März 1996 unter dem Titel Elektronik on-line. Während der folgenden Jahre erschienen diverse Beiträge, die meisten sind inzwischen inhaltlich überholt. Doch einige Veröffentlichungen haben dem Lauf der Jahre getrotzt. So stellten wir im Dezember 1997 Websites sowie Software vor, die den Bau von High-end-Lautsprecherboxen zum Thema hatten. Was hat sich hier in rund zwanzig Jahren getan?

Damals, als ich zu den engagierten Boxen-Entwicklern und Boxen-Bauern zählte, habe ich meine Boxen mit einigen gratis verfügbaren Programmen entwickelt und berechnet. Wohl aus diesem Grund kam mir der Gedanke, mich einmal umzusehen, was heute in der Szene geschieht. An Software gegen Geld ist in diesem Segment einiges auf dem Markt, doch audiophile Do-it-your-Selfer investieren lieber in die Boxen als in Software. Im ersten Anlauf war meine Recherche im Netz enttäuschend. Viele einschlägige Seiten wurden seit Jahren oder Jahrzehnten nicht mehr gepflegt. Die Programme von damals laufen heute, wenn überhaupt noch, nur in einer DOS-Box. Für die aktuellen Betriebssysteme wie Windows 7, 8 oder 10 ist die Auswahl mehr als mager. Das ist natürlich schade, zumal die physikalischen Grundlagen und die Methoden der Berechnungen bis heute im Wesentlichen unverändert geblieben sind.

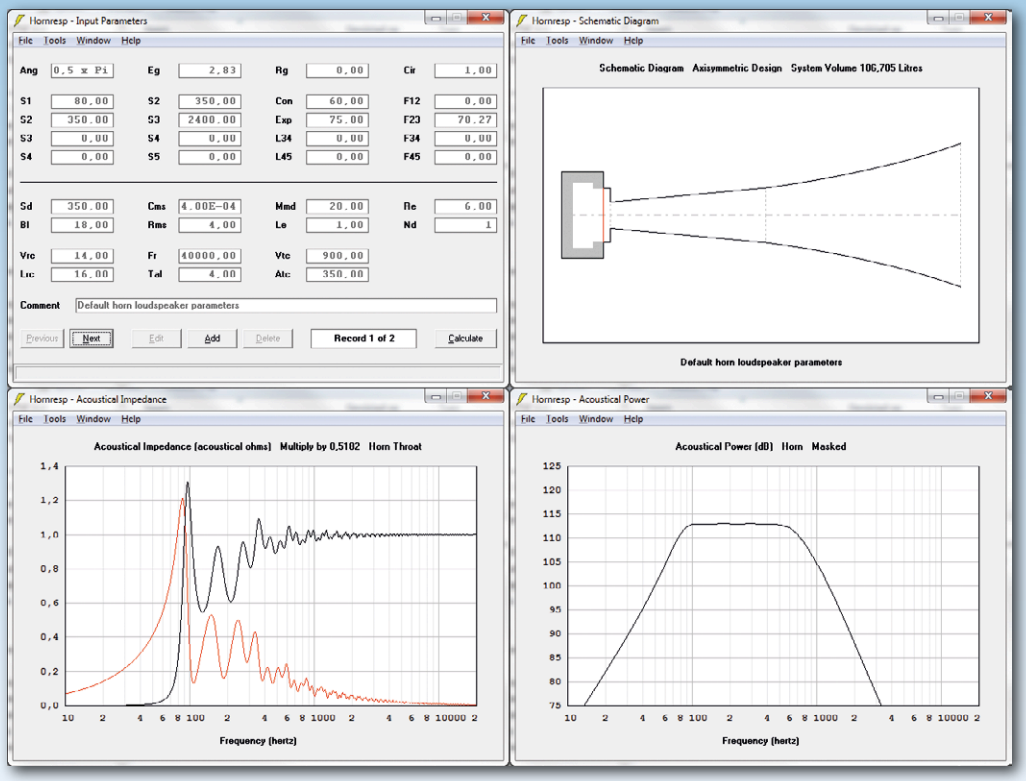
Zu den Programmen, die bereits 1997 existierten, gehört WinISD [1]. Dieses Programm wird auch heute noch gepflegt und fortentwickelt. Die Lettern *ISD* stehen für *Interactive Speaker Designer*, das Kürzel eines Programms, das Bassreflex- und Bandpass-Boxen berechnet. Darüber hinaus ist auch das Entwerfen und Simulieren passiver Filter und Equalizer möglich. Basis ist eine Datenbank, in der die meisten Hersteller erfasst sind. Für hochaktuelle Produkte müssen allerdings die nötigen Daten nachgetragen werden. WinISD ist ein leistungsstarkes Programm, es läuft klaglos, zumindest unter Windows 7.
Eine nicht minder interessante, seit ungefähr acht Jahren existierende Software ist Basta! von Tolvan Data [2]. Dieses Programm war zunächst kostenpflichtig, doch als der Autor vor Jahren die Weiterentwicklung einstellte, wurde es zur Freeware. Nach seinen Angaben ist das Programm auch unter Windows 8 oder 10 lauffähig. Basta! kann offene Schallleitsysteme, offene Boxen sowie Bassreflex- und Bandpass-Boxen berechnen und simulieren, und auch Ports und passive Radiatoren sind keine fremden Begriffe. Weitere Features sind das Kalkulieren der Schallbeugung an Gehäusekanten, das Berechnen von Geschwindigkeiten der Luft an Konussen und in der Umgebung von Ports, das Bestimmen der maximalen Konusauslenkungen und noch einiges mehr.

Für das Messen an Lautsprechern bietet die gleiche Website das Programm Sirp an [3], das mit einem logarithmischen Sinus-Sweep die Frequenzantwort von Treibern ermittelt und daraus die Impulsantwort herleitet. Die Methode ist zwar nicht so effizient wie das Messen auf der Basis von MLS zum Eliminieren der Umgebungseinflüsse, sie ist aber dennoch brauchbar. Sirp kann auch die Impedanzkurven von Systemen messen. Die von Sirp akquirierten Daten sind in Basta! importierbar.





Ein ebenso vielseitiges wie aktuelles Programm ist VituixCAD von Kimmo Saunisto [4], es läuft unter Windows 7, 8 oder 10. Mit dieser Software ist das Entwickeln und Simulieren von Mehrweg-Lautsprechersystemen möglich. Das Programm berechnet Boxen und Filter diverser Typen, darunter sind auch Boxen mit passiven Radiatoren sowie drei Zweikammergehäuse-Typen. Auch das Abstrahlverhalten und die Kantenbeugung werden vom Programm rechnerisch beherrscht. Ein Versuch, VituixCAD kennenzulernen, ist sicher der Mühe wert, auch wenn das Einarbeiten einigen Zeitaufwand erfordert.



Schließlich möchte ich noch die kostenlose Software Hornresp erwähnen, die insbesondere zum Entwerfen von Hornlautsprechern bestimmt ist. Die spartanisch gestaltete Download-Website [5] enthält nur wenige Textzeilen. Nach dem Download empfiehlt sich ein Besuch des Forums von Home Theater Shack [6], dort ist eine ausführliche Anleitung zum Einsatz von Hornresp zu finden. Fundierte Kenntnisse, zum Beispiel über Treiber und Hörner, sind für die erfolgreiche Arbeit unabdingbar. Außer den Hornlautsprechern können mit Hornresp auch andere Boxen-Typen wie geschlossene Boxen oder Bassreflex- und Bandpass-Boxen entwickelt werden. Das Programm existiert zwar schon recht lange, es wird jedoch von Autor David J. McBean immer noch auf dem Laufenden gehalten. Die aktuelle Version 40.20 ist unter Windows 7 problemlos lauffähig.

(160301)gd

Hornresp gehört zu den wenigen kostenfreien Programmen zum Entwickeln von Hornlautsprechern.

Weblinks

[1] www.linearteam.dk/

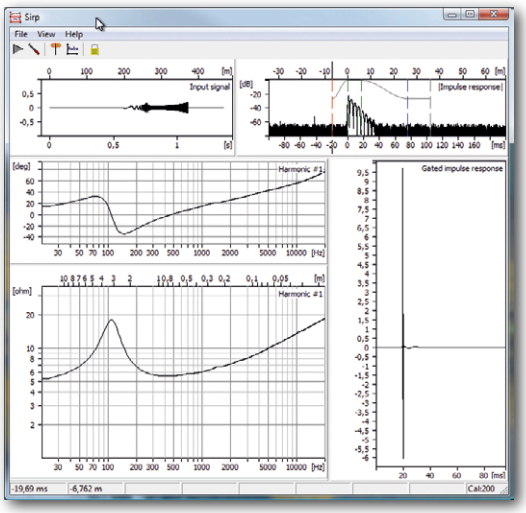
[2] www.tolvan.com/index.php?page=/basta/basta.php

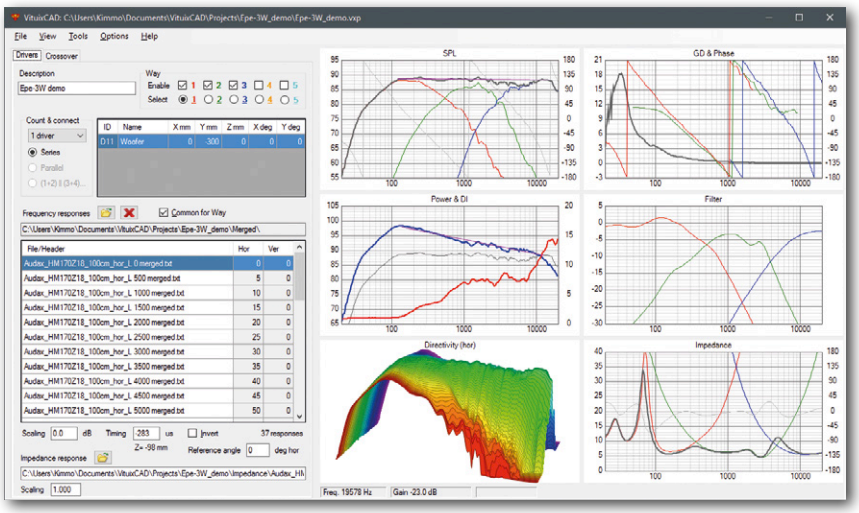
[3] www.tolvan.com/index.php?page=/sirp/sirp.php

[4] http://kimmosaunisto.net/Software/Software.html#VituixCAD

[5] www.hornresp.net/

[6] www.hometheatershack.com/forums/diy-subwoofers-general-discussion/36532-hornresp-dum-hmm-everyone.html

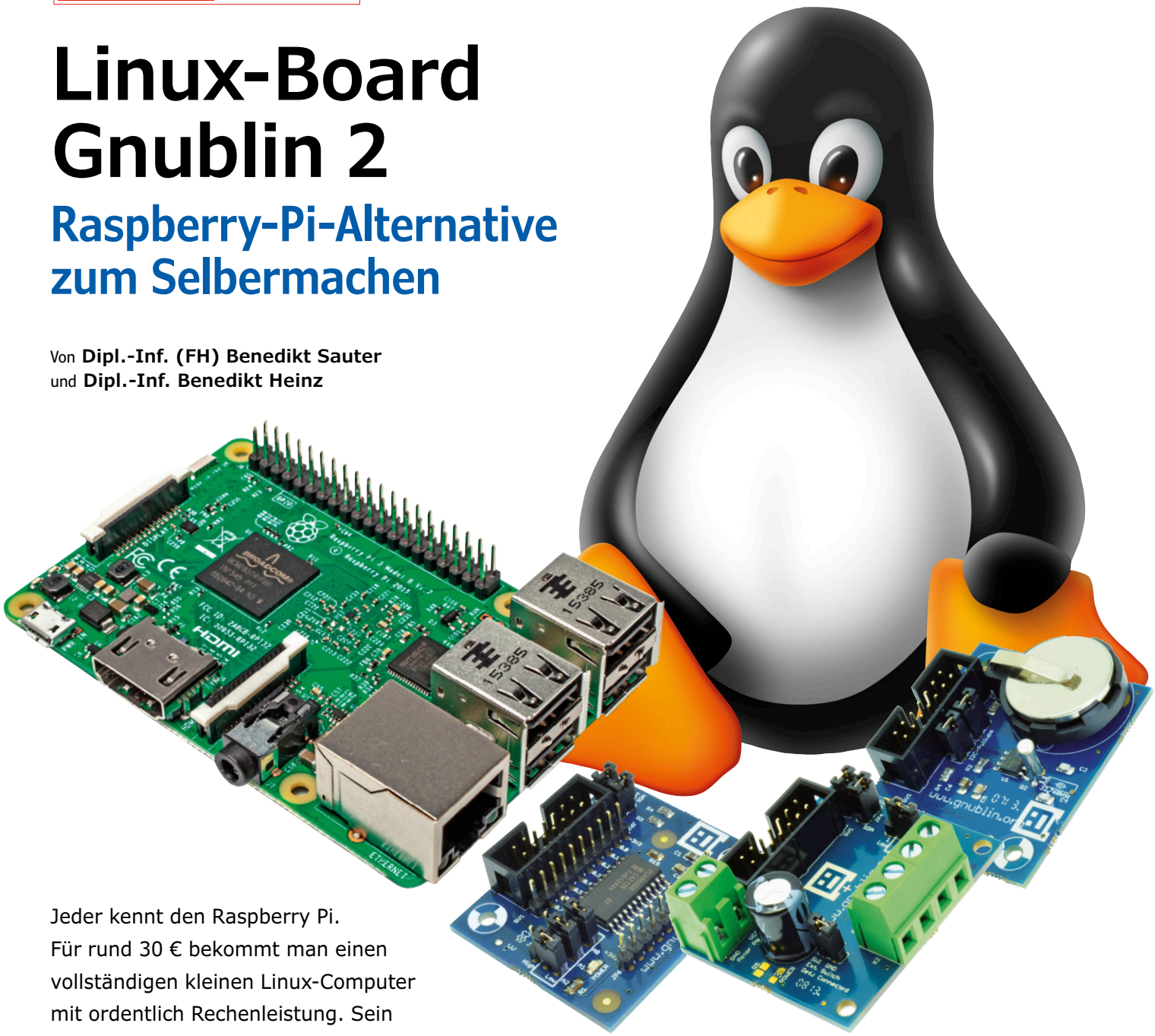




Linux-Board Gnublin 2

Raspberry-Pi-Alternative zum Selbermachen

Von **Dipl.-Inf. (FH) Benedikt Sauter**
und **Dipl.-Inf. Benedikt Heinz**



Jeder kennt den Raspberry Pi. Für rund 30 € bekommt man einen vollständigen kleinen Linux-Computer mit ordentlich Rechenleistung. Sein Schönheitsfehler: Er ist zwar sehr preiswert, doch sein Hardwaredesign ist nicht allzu offen. Deshalb haben die Autoren einen eigenen Single-Board-Computer mit aktueller ARM-Cortex-A5-CPU entworfen. Er kann vielleicht preislich nicht voll konkurrieren, doch ist seine völlig offene Soft- und Hardware beliebig anpass- und erweiterbar. So kann man sich ein Linux-Board nach Maß selbst bauen, für die Lehre, die Entwicklung und den industriellen Einsatz.

Gnublin 2 hat natürlich eine Vorgeschichte, denn wo eine 2 steht, gab es auch einmal einen Vorgänger. Gnublin 1 war die Basis für das Elektor Linux-Board und einen Embedded-Linux-Kurs in Elektor [1], der etliche weitere Artikel nach sich zog und für den im Elektor-Store immer noch eine ganze Reihe Hardware angeboten wird [2]. Unsere Firma *embedded projects* hat dafür eine eigene Webseite ins Netz gestellt [3].

Mittlerweile sind rund fünf Jahre ins Land gegangen – in der Mikroelektronik eine kleine Ewigkeit. In der Zwischenzeit erschien nicht nur der RPi mit allen seinen Nachfolgern und Konkurrenten; es kommen auch immer wieder neue Linux-Boards auf den Markt, die im Grunde die gleiche Zielgruppe abdecken und immer besser werden. Die Frage also ist: Macht eine neue Version von Gnublin überhaupt Sinn? Wenn die Antwort „ja“ lauten soll, dann muss es mehr sein als nur ein „me

too“. Außerdem erfreut sich Gnublin 1 immer noch großer Beliebtheit. Das liegt unserer Meinung nach daran, dass folgende Aspekte bei vielen anderen Boards zu kurz kommen:

- Gnublin 1 ist komplett offen. Es sind Schaltplan, Platinen-Layout und Software verfügbar.
- Die verwendeten Bauteile sind auch in kleinen Stückzahlen erhältlich und die „nur“ sechslagige Platine ist selbst in kleinen Stückzahlen vergleichsweise kostengünstig.
- Statt wie die meisten neuen Linux-Boards mit möglichst hoher Leistung zu glänzen, überzeugt das Gnublin 1 mit einer äußerst geringen Energieaufnahme. Viele eingebetteten Systeme brauchen keine aufwendige 3D-Grafik. Bei Dauerbetrieb ist ein geringer Energieverbrauch wünschenswert.

Genug Gründe für uns, den Markt nach geeigneten ARM-SoCs zu scannen, als Basis für eine Version 2 des Gnublin-Boards. Dabei gab es zwei wichtige Kriterien: Geringer Energieverbrauch und optimale Linux-Unterstützung.

SoC-Auswahl

Da der Energieverbrauch zentral ist, lag unser Fokus auf Single-Core-SoCs der Typen Cortex-A5 und Cortex-A7. Die Wahl fiel dabei schließlich auf den ATSAMA5D41 [4] von Atmel mit Cortex-A5-Kern. Bezüglich Linux-Unterstützung hatten wir mit dem Gnublin 1 und weiteren Projekten bereits recht viel Erfahrung, und dabei auch recht frustrierende Erlebnisse gehabt. Dies gilt insbesondere für den LPC3131 von NXP des Gnublin 1. Die letzte brauchbare Kernelversion für dieses SoC ist 2.6.33. Sie ist mehr als sechs Jahre alt! Obwohl der LPC3131 noch als aktiv bzw. „in production“ gelistet ist, wird dieser Chip seit Jahren schon nicht mehr im Linux-Kernel unterstützt.
Ganz anders sieht es hier bei Atmel aus. Fast alle Atmel-SoCs – auch etliche Jahre alte Exemplare – werden heute noch unterstützt. Es sind nicht einmal spezielle Patches des Herstellers nötig, denn der offizielle Linux-Kernel selbst bietet Unterstützung für diese SoCs. Atmel stellt zudem ausführliche Dokumentation ohne NDA (Non Disclosure Agreement = Verpflichtung zur Geheimhaltung) zur Verfügung und arbeitet bei der Entwicklung eng mit der Linux-Community zusammen. Folglich pflegt die Entwicklergemeinde des Kernels den Code für diese ICs mit, wenn Änderungen im Kernel vorgenommen werden. Das ist die beste Grundlage für eine langfristige Unterstützung der SoCs durch aktuelle Linux-Kernel.
Weiter bietet der ATSAMA5D41 einige Möglichkeiten, die anderen Linux-Boards fehlen. Beispielsweise stehen zwei serielle Schnittstellen zur Verfügung – eine für die Linux-Konsole und eine weitere für benutzerspezifische Anwendungen. Zudem lässt sich Gnublin 2 durch den High-speed-Device-Port direkt via USB mit einem PC verbinden – der Raspberry Pi kann das zum Beispiel nicht. Bei Embedded-Systemen wird oft eine Echtzeituhr benötigt, die auch nach einem Spannungsausfall die korrekte Zeit liefert. Bei Gnublin 2 lässt sich die Echtzeituhr mit einer zusätzlichen Batterie einfach puffern. Auch ein resistiver Touchscreen lässt sich direkt, ohne externe Spannungsversorgung anbinden.
Genug Gründe für eine Version 2 von Gnublin auf der Basis dieses Chips. Gnublin 2 ist wie sein Vorgänger als Basis für eigene Entwicklungen bzw. eigene Produkte gedacht. Neben den üblichen Daten werden deshalb sogar Messprotokolle aus dem EMV-Labor zur Verfügung gestellt.

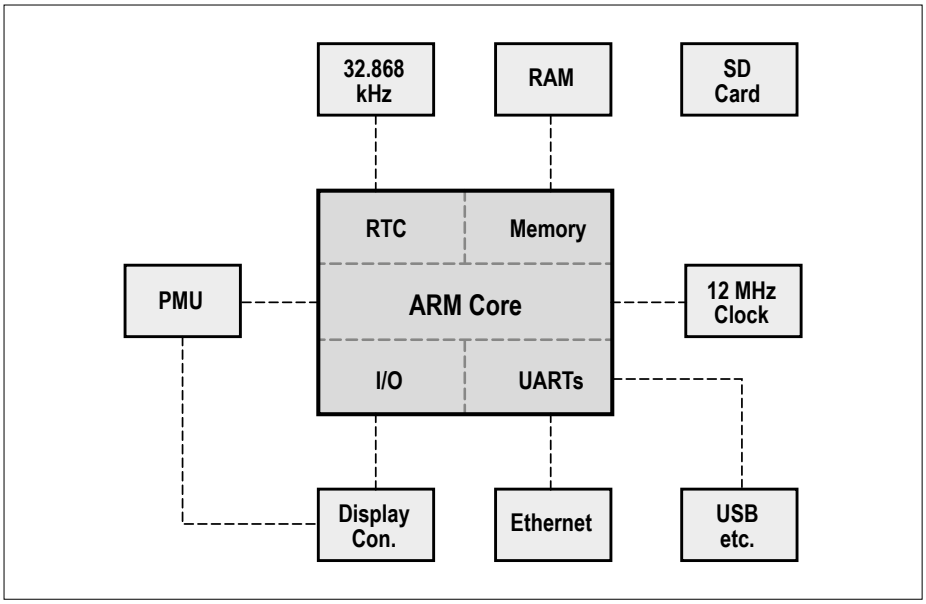


Bild 1. Das Blockschaltbild von Gnublin 2 zeigt das fett umrandete SoC mit Subeinheiten sowie die extern angeschlossene Hardware nach Funktionen.

Einsatzmöglichkeiten

Gnublin 2 ist überall da zuhause, wo es um Steuerungen und Messtechnik geht, bzw. ein Embedded-System für eine Anwendung oder ein Produkt, bei dem der klassische Mikrocontroller durch eine linux-taugliche CPU mit genug Leistung ersetzt werden soll. Die Grundschaltung inklusive Prozessor, Speicher und Stromversorgung kommt je nach Stückzahl auf Preise von lediglich 15...20 €. Gnublin 2 eignet sich daher auch für kleinere Produktserien.
Dank offener Stückliste, Schaltung und Platinen-Layouts sowie EMV-Messungen und Kalkulationstools spart man sich gerade bei Kleinserien eine Menge Aufwand, den man sonst bei der Anpassung „fertiger“ Boards wie RPi und Varianten hätte.
Außerdem ist die Anbindung von Zusatz-Hardware dank der Dokumentation kein Hexenwerk. Beim üblichen Entwicklungsprozess vom Prototypen über handbestückte Muster gelangt man zudem nur durch das EMV-Labor zur ersten Serie. Dass solche Messungen schon vorliegen, ist also eine willkommene Abkürzung. Und selbst bei Einzelstücken, bei denen ja der Aufwand von EMV-Überprüfungen unangemessen wäre, ist es ein gutes Gefühl, zu wissen, woran man hier bezüglich HF-Verhalten ist.

Organisation und Bauteile

Erfreulicherweise benötigt GNUBLIN 2 im Idle-Zustand des Linux-Systems weniger als 0,3 W – und das bei einem Takt von 600 MHz sowie immerhin 256 MB RAM. In der Blockschaltung von **Bild 1** ist das zentrale SoC fett umrandet. Man erkennt, wie neben dem eigentlichen Kern seine

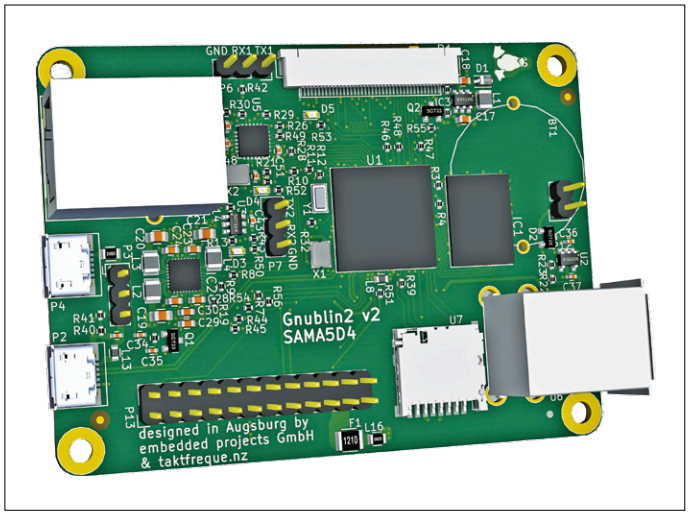


Bild 2. 3D-Rendering der Platine von Gnublin 2.

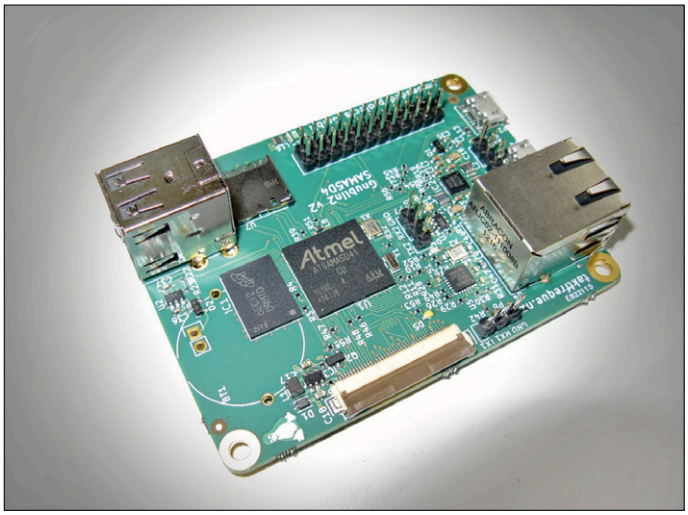


Bild 3. Der fertige Prototyp von Gnublin 2.

wichtigsten Subeinheiten wie die RTC (Real Time Clock), der Speicher etc. mit externer Hardware verbunden sind. Einen detaillierten Schaltplan kann man unter [6] herunterladen, er ist zu groß, um ihn hier abzudrucken.

Der 256-MB-DDR2-Speicher (IC1) ist mit 16 bit an das SoC ATSAMA5D41 (U1) angebunden. Dies erlaubt im Vergleich mit anderen Boards, die hierfür einen 32 bit breiten Bus vorsehen, eine einfachere und günstigere Platine mit geringerer Leistungsaufnahme der Elektronik. Dank DDR2 ergibt sich bei einem RAM-Takt von 300 MHz immer noch eine Datenrate von über 1 GB/s.

Das SoC benötigt verschiedene Spannungen für I/O und den Prozessorkern. Diese Spannungen werden von der PMU (Power Management Unit) des Typs ACT8865 (IC2) [5] bereitgestellt. Diese PMU beherbergt mehrere Schalt- und Linearregler und kann via I²C vom SoC konfiguriert werden.

Die Versorgung der im SoC integrierten Echtzeituhr übernimmt der Low-drop- und Ultra-low-power-Linearregler TPS78001 (U2), der aus einer Knopfzelle oder über eine Stiftleiste extern gespeist wird. Zur Echtzeituhr gehört natürlich auch ein Uhrenquarz mit den typischen 32,768 kHz (Y1). Der normale Takt wird von einem 12-MHz-Quarz (X1) abgeleitet.

Im Vergleich zu Gnublin 1 kann die zweite Version jetzt direkt ein TFT-Display

Mit Display und Browser im Internet surfen

Viele verschiedene Linux-Distributionen lassen sich mit dem Cortex-A5-Board einsetzen. Wir selbst nutzen zur Entwicklung zwar vorwiegend Debian Jessie [1], doch Yocto oder Ubuntu sind ebenfalls geeignet. Debian ist in der Basisversion jedoch deutlich schlanker als Ubuntu. Nichtsdestotrotz verfügt Debian über eine enorme Anzahl von Softwarepaketen, die sich via Netzwerk bequem mit dem *apt-Tool* nachinstallieren lassen. Bereits installierte Pakete lassen sich auch automatisch über das Netzwerk aktualisieren. Dies ist beispielsweise für Sicherheitsupdates sehr hilfreich und wünschenswert. Auch der GNU-C-Compiler lässt sich via apt installieren.
Software kann dann auch direkt auf dem Board kompiliert werden, sofern der Code nicht zu umfangreich ist. Eine einfache grafische Oberfläche steht mit LXDE [2] zur Verfügung. Diese ist sehr ressourcenschonend und lässt sich mit „apt-get install lxde“ installieren. Zur Darstellung von sehr aufwendigen Websites reichen die verfügbaren 256 MB RAM heute teilweise nicht mehr aus, aber mit Midori [3] steht ein schlanker Webbrowser zur Verfügung, der viele Websites mit HTML5, CSS und Javascript darstellen kann. Dieser Browser lässt sich mit „apt-get install midori“ installieren.

Das Bild beweist, dass Gnublin 2 ohne Probleme normale Webseiten aufrufen und darstellen kann.

[1] Jessie: www.debian.org/releases/jessie
[2] LXDE: http://wiki.lxde.org/de/LXDE_Debian
[3] Midori: http://midori-browser.org

ansteuern. Das ermöglicht viele neue Anwendungen. Damit das Board trotz Display mit einer einzigen 5-V-Spannung auskommt, gibt es noch den Aufwärtsregler AP5724 ICs (IC3). Dieser versorgt die LED-Hintergrundbeleuchtung des Displays mit einem konstanten Strom von 40 mA bei rund 24 V. Durch den integrierten ADC ist das SoC in der Lage, resistive Touchscreens direkt anzubinden. Per Ethernet-Phy KSZ8081RNA/D (U5) kann das Board direkt mit 10/100 MBit/s in Netzwerke eingebunden werden. Der nötige Koppeltrafo ist dabei direkt in der RJ45-Buchse (U4) untergebracht. Auch für eine valide MAC-Adresse ist mit Hilfe des via I²C angebundenen EEPROMs 24AA025E48 (U3) gesorgt.

Wie bei fast jedem anderen Linux-Board ist auch bei Gnublin 2 ein Massenspeicher vorhanden: Bootloader und Linux-System werden beim Start direkt von einer MicroSD-Karte (U7) geladen.
Das SoC verfügt insgesamt über drei High-speed-USB-Ports. Zwei davon sind als Host-Ports über die USB-Doppelbuchse U6 verfügbar. Der dritte Port steht über die Micro-USB-Buchse P2 als Device-Port zur Verfügung. Auf diese Weise kann sich das Board an einem Rechner als nahezu beliebiges USB-Gerät ausgeben. Man kann so z.B. eine USB-Netzwerkverbindung zwischen Host und Gnublin 2 aufbauen. Die Spannungsversorgung des Boards kann dabei direkt über die Micro-USB-Buchse P2 erfolgen. Alternativ steht eine zweite Micro-USB-Buchse (P4) als gesonderter Anschluss für eine leistungsfähigere Stromversorgung (etwa per Smartphone-Netzteil) zur Verfügung. Das ist dann hilfreich, wenn mehrere USB-Geräte an Gnublin 2 angeschlossen sind. Der Jumper P3 wählt aus, ob das Board über die Buchse P4 oder P2 versorgt werden soll.
Die I/O-Pins des SoCs arbeiten mit 3,3-V-Pegel. Viele davon lassen sich über die Stiftleiste P13 abgreifen. Hier steht wie schon bei Gnublin 1 ein SPI-Bus mit einigen Chip-Select-Ausgängen zur Verfügung. Auch ein I²C-Bus und der zweite serielle UART des SoCs sind über P13 und zusätzlich via P7 verfügbar. Dieser UART wird vom bestehenden Linux-System nicht als Konsole genutzt. Er steht also exklusiv für benutzerspezifische Anwendungen zur Verfügung. P6 schließlich bietet Zugang zum ersten UART, der eine Linux-Konsole zur Verfügung stellt und beim Start die Kernelmeldungen ausgibt.

Crowdfunding!

Für dieses Projekt haben embedded projects und Elektor eine Crowdfunding-Kampagne gestartet. Wer sich dort beteiligen bzw. ein Gnublin-2-Board erwerben möchte, findet einen Weblink unter: **www.gnublin.org**

Auch der Bootloader lässt sich über diesen UART steuern.

Bild 2 zeigt ein 3D-Rendering der mit KiCAD erstellten Platine. Sowohl die Layout-Dateien als auch der zugehörige Schaltplan, die Software, die Dokumentation und weitere Infos stehen unter [6] zum kostenlosen Download zur Verfügung. **Bild 3** zeigt ein Foto der fertigen und bestückten Platine. Im **Kasten** findet man die grundlegenden Infos, wie man auf der Basis von Gnublin 2 plus Display die Software für einen kompletten Linux-Computer zusammenstellt.

Ausblick

Dieser Artikel enthält eine knappe Beschreibung dessen, um was es sich bei Gnublin 2 dreht. Mit Hilfe der Materialien von [6] kann man schon prima eigene Projekte realisieren. Auf der anderen Seite haben sich im Laufe der Entwicklung so interessante Teilaspekte ergeben, dass diese jeweils eigene Artikel verdient haben. Rechnen Sie also demnächst mit einer detaillierten Beschreibung der Entwicklung einer Platine mit der Layout-Software KiCAD. Und zwar nicht am Beispiel einer Drei-Bauteile-Platine, sondern aussagekräftig anhand des Gnublin-2-Boards. Und wenn hier schon Messungen bezüglich der Störstrahlung vorgenommen wurden, wird es auch einen Artikel über EMV geben. Sie ahnen schon, was hier als Beispiel dienen wird. Last not least wird auch noch ein Artikel kommen, der Schritt für Schritt zeigt, wie man eigene Varianten des Linux-Boards entwickelt. ◀

(160090)

Über die Autoren

Benedikt Sauter ist Diplom-Informatiker. Nach dem Studium beschäftigte er sich mit diversen Mikrocontroller-Projekten und schlug direkt den Weg in die Selbständigkeit ein. Alsbald folgte die Gründung der *embedded projects GmbH*. Er hat sich schon in jungen Jahren leidenschaftlich mit Soft- und Hardware beschäftigt und auch in Elektor schon einige Open-Source-Projekte veröffentlicht.
Dipl.-Inf. Benedikt Heinz entwickelt seit über 15 Jahren Digitalelektronik und Software für Mikrocontroller und CPUs. Aufgrund der langjährigen Linux-Nutzung auf dem Desktop setzt er auch vorwiegend Open-Source-Software ein und entwickelt diese auch selbst weiter. Derzeit arbeitet er an Software, mit der sich aufwendige Boards (PCIe, DDR-RAM etc.) einfacher mit KiCAD layouten lassen.

Weblinks

[1] Embedded Linux leicht gemacht: www.elektormagazine.de/120026

[2] Gnublin-Produkte: www.elektor.de/devtools/gnublin-linux

[3] Gnublin-Webseite: http://gnublin.embedded-projects.net

[4] ATSAMA5D41: www.atmel.com/devices/ATSAMA5D41.aspx

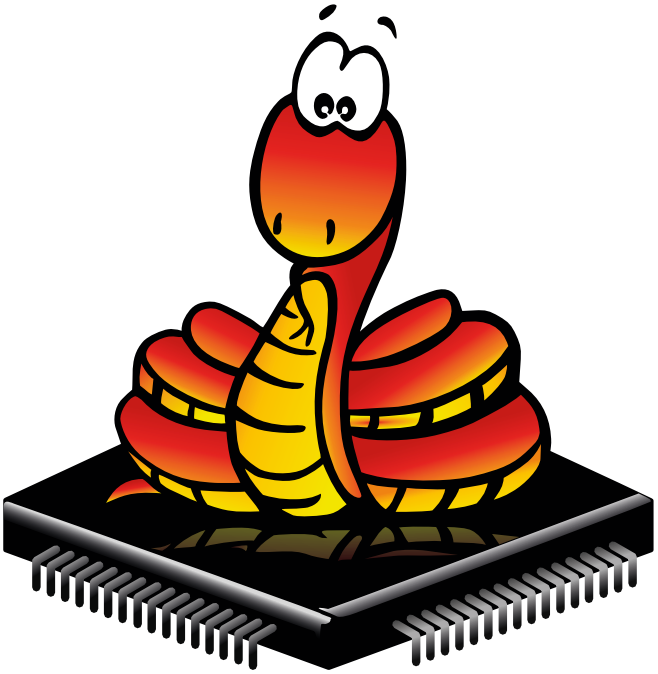
[5] ACT8865: www.active-semi.com/products/power-management-units/act88xx//

[6] Gnublin-2-Materialien: www.gnublin.org/index2.html

MicroPython

Python für kleine Systeme

Von **Dogan Ibrahim** (Großbritannien)



MicroPython ist eine hocheffiziente und leistungsfähige, von Python abgeleitete Programmiersprache mit einer kleinen Sammlung von Bibliotheken. MicroPython benötigt nur 256 KB Programmspeicher und 16 KB RAM, so dass es gut in Mikrocontrollern und anderen eingebetteten Systemen mit beschränkten Ressourcen eingesetzt werden kann.

Python ist weltweit an vielen Universitäten und Fachhochschulen als erste Programmiersprache für Studenten im Einsatz. Die große Sammlung von leistungsstarken Bibliotheken und die Benutzerfreundlichkeit von Python machen sie zur idealen Sprache für alle Programmierneulinge. MicroPython, ursprünglich für 32-Bit-ARM-Mikrocontroller entwickelt, ist mit Python kompatibel und stellt wegen seiner geringen Größe eine hervorragende Wahl für den Einsatz auf Embedded-Prozessoren dar. Mit MicroPython kann man einen komplexen und dennoch handlichen Code entwickeln, um statt mit Sprachen wie C oder C++ eingebettete Systeme zu steuern. MicroPython ermöglicht sowohl erfahrenen Programmierern als auch Einsteigern, kleine eingebettete Systeme zu programmieren.

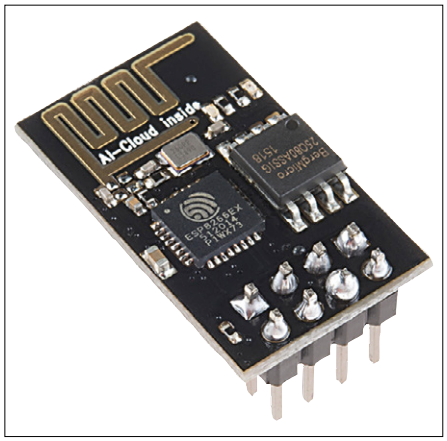


Bild 1. Der ESP01 ist ein äußerst praktisches und populäres ESP8266-Modul.

Bild 2. Der BBCmicro:bit kann online in MicroPython programmiert werden.

Die Hauptgründe, hier nicht direkt Python 3 einzusetzen, sind der Mangel an ausreichendem Speicher und das Fehlen von Hard- und Softwarefunktionen wie Multitasking und Multiprocessing bei eingebetteten Prozessoren. Die Unterschiede zwischen Python 3 und MicroPython werden in [1] erläutert.

Warum MicroPython?

MicroPython bietet (wie Python) im Vergleich zu anderen Programmiersprachen einige einzigartige Funktionen:

- (Micro-)Python ist interaktiv. Das bedeutet, dass das Programm nicht kompiliert und in den Zielprozessor geladen, sondern zur Laufzeit abgearbeitet und interpretiert wird. Obwohl dies die Programme etwas langsamer macht, bietet es dem Anwender den Vorteil, einfach mit dem Code experimentieren zu können. Man kann interaktive Berechnungen wie mit einem Taschenrechner durchführen oder an Teilen des Programms so lange herumschrauben, bis das gewünschte Ergebnis erreicht ist. Dieses Read-Evaluate-Print-Loop (REPL) genannte Vorgehen steht bei kompilierten Sprachen wie C oder C++ nicht zur Verfügung.
- Neben der großen Anzahl integrierter Funktionen ist ein umfangreicher Satz von Funktionsbibliotheken verfügbar, die in ein Programm eingebunden werden können. Es gibt beispielsweise Bibliotheken für die Erzeugung von Zufallszahlen, zu trigonometrischen Berechnungen, zum Musizieren, für Netzwerkfunktionen, zur Zeichenfolgen- und Dateiverarbeitung, Grafik und Spiele und vieles mehr.
- MicroPython kann mit anderen Programmiersprachen wie C oder C++ vermischt werden. Dies erhöht die Leistungsfähigkeit und Flexibilität, da Teile des Codes, die eine hohe Geschwindigkeit erfordern, in besser geeigneten Sprachen entwickelt werden können.
- Es werden Ausnahmen und eine Fehlerbehandlung unterstützt, was bei Echtzeitprogrammierung besonders wichtig ist. Ohne eine angemessene Fehlerbehandlung kann sich ein fehlerhaftes Programm in einem unbekannten Zustand festfahren, was sehr negative Folgen haben kann.
- MicroPython ist Open Source. Man braucht lediglich die neueste Version herunterzuladen [2] und auszuführen, alles ohne Kosten. Der Quellcode von MicroPython kann modifiziert und auf bestimmte Prozessoren portiert werden.
- Schließlich ist die Sprache mit ihrer leicht zu erlernenden Syntax auch vom Menschen lesbar.

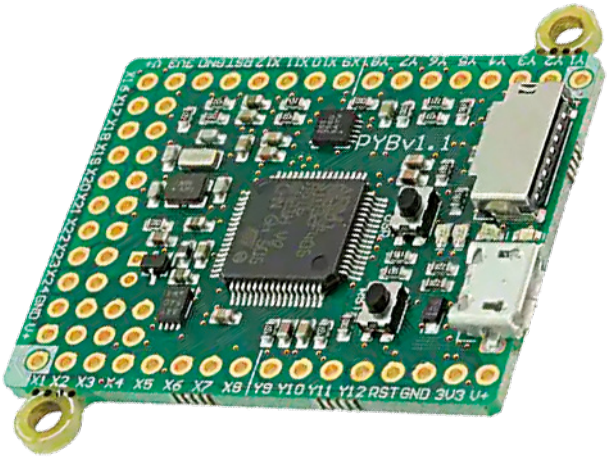


Bild 3. Das Pyboard ist das offizielle MicroPython-Mikrocontrollerboard.

Bild 4. WiPy ähnelt dem ESP8266-basierten NodeMCU-Board, ist aber mit einem CC3200 ARM Cortex M4 ausgestattet.

Was MicroPython kann und was es nicht kann

MicroPython kann eigentlich alles, was andere Programmiersprachen auch können: Hardware wie LEDs und Displays, Schalter und Tasten, Sensoren, Motoren und so weiter steuern. Kommunikationsbusse wie RS232, CAN, I²C, SPI und andere werden dank eingebauter oder externer Bibliotheken problemlos genutzt. Netzwerk- und WLAN-basierte Programme zur Kommunikation mit anderen Netzwerkgeräten (auch in IoT-Systemen) können leicht geschrieben werden.

Da MicroPython eine Interpreter-Sprache ist, ist sie im Vergleich zu anderen eingebetteten Programmiersprachen langsamer und daher keine gute Wahl für schnelle digitale Signalverarbeitung oder Echtzeitanwendungen mit hoher Ausführungsgeschwindigkeit. Außerdem – aber das ist heutzutage nicht mehr ganz so wichtig - benötigt MicroPython etwas mehr Speicherplatz als die meisten anderen eingebetteten Sprachen. Da MicroPython nicht Python ist, werden auch nicht alle Python-Bibliotheken unterstützt. So kann es passieren, dass ein in Python entwickeltes Programm nicht auf einem Embedded-System mit MicroPython funktioniert.

Boards mit MicroPython-Unterstützung

Die Zahl der Entwicklungsboards, die MicroPython unterstützt, nimmt stetig zu.

ESP8266

Boards, die mit dieser beliebten WLAN-fähigen MCU (**Bild 1**) mit TCP/IP-Stack und USB-Schnittstelle ausgestattet sind, können in MicroPython programmiert werden. Die MCU basiert auf einer 32-Bit-RISC-CPU. MicroPython unterstützt GPIO, SPI, I²C, UART, ADC und I²S. Die entsprechenden Boards sind sehr preisgünstig und ideal, wenn Sie mit MicroPython experimentieren möchten.

BBCmicro:bit

Dieses kreditkartengroße Board (**Bild 2**) stellt viele Funktionen wie 25 LEDs, zwei Drucktaster, einen Beschleunigungssensor, einen Kompass, GPIO, I²C, UART und ADC zur Verfügung und wird von MicroPython unterstützt. Programme können online erstellt werden, man benötigt nicht einmal einen Hardware-Aufbau. Das BBCMicro:bit ist sehr empfehlenswert für Einsteiger in Sachen Programmierung.

Pyboard

Das Pyboard (**Bild 3**) basiert auf dem Cortex-M4-Prozessor der Serie STM32F und wird mit vorinstalliertem MicroPython geliefert. Das Pyboard kann über seinen USB-Anschluss mit dem PC verkabelt werden. Das Board verfügt über eine Echtzeituhr, einen Beschleunigungssensor, GPIO, ADC, vier LEDs und einen microSD-Kartenslot.

WiPy

Wie das Pyboard ist auch WiPy (**Bild 4**) von Haus aus mit MicroPython ausgestattet. Die Pins sind zur Aufnahme auf einem Steckboard geeignet. WiPy basiert auf dem Prozessor CC3200 Cortex M4, läuft mit 80 MHz und besitzt UART, SPI, I²S, ADCs, WLAN, GPIO, Timer und Hash-/Encryption-Module.

Weitere interessante Embedded-Entwicklungsboards mit MicroPython-Unterstützung sind: Teensy 3.x, SAMD21, LoPy, STM32F4-Discovery, Raspberry Pi und BeagleBone (beide mit dem vollständigen Python-3-Code).

Programmbeispiel in MicroPython

Hier ein einfaches MicroPython-Programm, das auf dem BBCmicro:bit läuft. Es handelt sich um ein Thermometer, bei dem kontinuierlich die CPU-Temperatur gelesen und eine entsprechende Meldung auf der LED-Matrix angezeigt wird. Wenn die Temperatur 25 °C überschreitet, wird die Meldung „HIGH“ angezeigt, liegt die Temperatur zwischen 20 °C und 25 °C, erscheint „MEDIUM“, ansonsten „LOW“.

```
#Simple CPU thermostat program
from microbit import *

while True:
  temp = temperature()
  If temp >= 25:
    display.scroll("HIGH")
  elif temp >= 20 and temp < 25:
    display.scroll("MEDIUM")
  else:
    display.scroll("LOW")
```

(160315)

Weblinks

[1] https://github.com/micropython/micropython/wiki/Differences

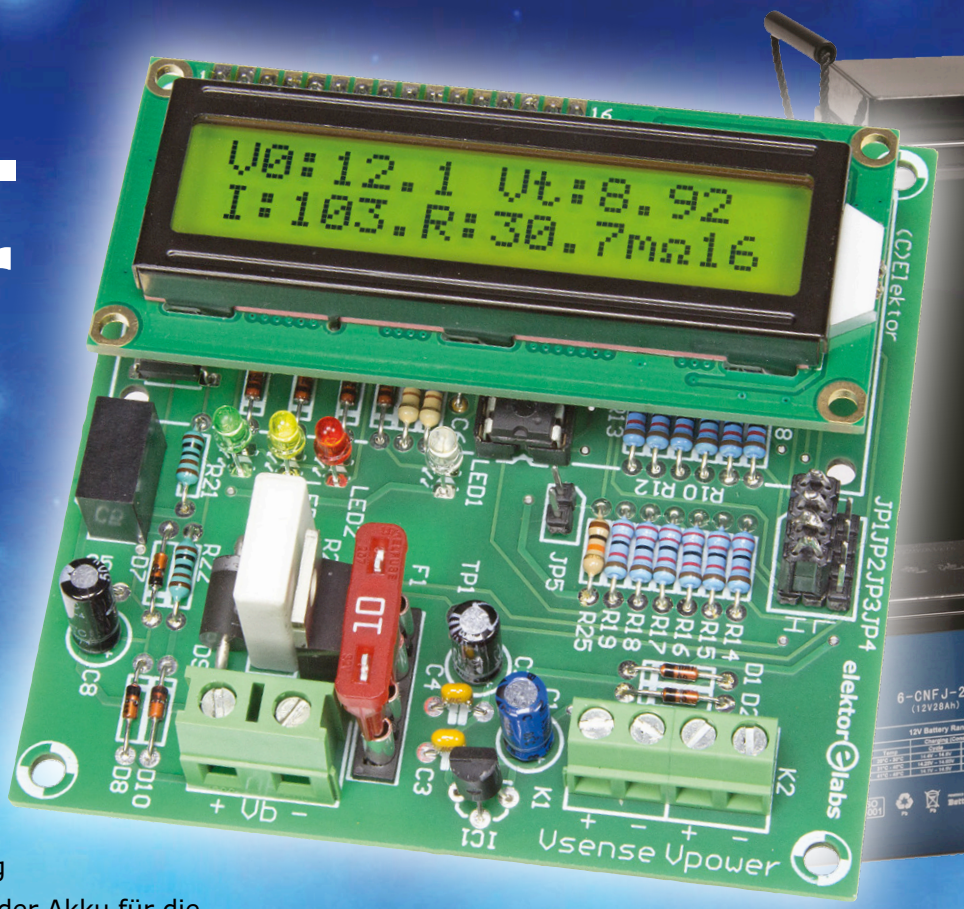
[2] https://github.com/micropython/

[3] http://micropython.org

e LABS PROJECT

Bleiakku-Aktivator 0...30 V

Akku-Zustand im Blick



Seit Jahren vertreibt ein bekannter Elektronik-Sortimenter einen unkomplizierten „Batteriewächter“ [1], mit dem Blei-Akkus tatsächlich länger gebrauchsfähig sind. Das Prinzip ist wenig spektakulär: In Abständen von 30 s wird der Akku für die Dauer von 100 µs impulsartig mit etwa 100 A belastet. Wie auch der Zustand eines Akkus gemessen werden kann, verraten wir in diesem Beitrag.

Von **Jan Lichtenbelt** (NL)

INFOS ZUM PROJEKT

Messen & Testen
Hobby | Modellbau
Akkus

Einsteiger
➡ Fortgeschrittener
Experte

Etwa 3 Stunden

Kein spezielles Werkzeug nötig

Rund 60 €

Das Verfahren stützt sich auf die Theorie [2], dass kurzzeitige hohe Stromimpulse einer Sulfatierung der Akku-Bleiplatten entgegenwirken. Beim Autor ist der „Batteriewächter“ von Conrad einem simplen Ladegerät parallel geschaltet, das mit einem LM317 als Spannungsregler arbeitet. Der LM317 ist auf 13,8 V eingestellt, der Blei-Akku hat die Kapazität 7 Ah. Die hohen Stromimpulse haben keine erkennbaren Auswirkungen auf die Arbeitsweise des LM317.

Nicht nur der Sulfatierung wird entgegengewirkt, die Höhe der Spitzenströme gibt auch Aufschluss über die Qualität des Akkus. Wenn ein hoher Strom kurzzeitig durch einen externen Shunt fließt und der Abfall der Klemmenspannung gemessen wird, lässt sich der Innenwiderstand des Akkus bestimmen:

Ri = (V0-Vt) / I, wobei

I = ΔVshunt / Rshunt ist.

Die unbelastete Klemmenspannung ist hier mit V0 bezeichnet, und Vt ist die belastete Klemmenspannung. Der Innenwiderstand Ri gilt als Maß für den Akku-Zustand, er muss möglichst niedrig sein. Unser nachfolgend beschriebener Akku-Aktivator ist für Blei-Akkus mit den Nennspannungen 2 V, 6 V, 12 V und 24 V ausgelegt. Mit ihm können nicht nur Blei-Akkus geprüft und aktiviert werden, auch das Messen der Innenwiderstände anderer Akkus und auch Batterien bis 30 V ist möglich.

Gedacht hatten wir zuerst an eine Erweiterung der Variante von Conrad, doch dies hätte zu einer Konstruktion geführt, die wir mit den hohen Strömen nicht für vereinbar halten. Außerdem scheint jene Ausführung das Handicap zu haben, dass der Shunt-Widerstand 0,1 Ω/2 W nicht für hohe Impulsbelastung konstruiert ist. Bei häufigem Einsatz kann der Shunt unbemerkt defekt werden.

In der Literatur [3] ist dargelegt, dass der Innenwiderstand in der Regel stark von der Temperatur, dem Ladezustand (SOC, State of Charge) und dem Alter eines Akkus oder einer Batterie abhängt. Mit sinkender Temperatur steigt der Innen-

widerstand, gleiches gilt für die Alterung und auch die Tiefentladung. Wenn die Abhängigkeit des Innenwiderstands von einem dieser drei Parameter gemessen werden soll, müssen die übrigen Parameter konstant gehalten werden. Vergleichsweise hohe Werte bei Zimmertemperatur sind in der Regel ein Indiz für einen schlechten Ladezustand oder ein hohes Alter. Der Innenwiderstand ist neben der Klemmenspannung ein wichtiges Kriterium für die Gebrauchsfähigkeit eines Akkus.

Aktivator für 2 V, 6 V, 12 V oder 24 V

Wie die Schaltung in Bild 1 zeigt, haben wir unseren Bleiakku-Aktivator mit einem Mikrocontroller realisiert. Der Mikrocontroller generiert zyklisch im Abstand von 30 s einen 100 µs breiten Impuls, er wird dem Gate eines MOSFETs zugeführt. Der MOSFET schaltet den Shunt, der an den Akkuklemmen liegt. Eine Diode schützt die Schaltung bei versehentlicher Verpolung des Akkus. Wenn die Akku-Nennspannung 6 V beträgt, resultiert aus dem Wert 50 mΩ des Shunts ein theoretischer Spitzenstrom von ungefähr 100 A.

Die Klemmenspannung wird über separate Leitungen gemessen, so dass die hohen Ströme die Messungen nicht verfälschen (Vierpunkt-Messung). Nur an den Akkuklemmen sind beide Stromkreise miteinander verbunden. Die Höhe des Stroms wird aus dem Spannungsabfall am Shunt hergeleitet.

LED1, eine blaue LED, leuchtet im Rhythmus der vom Mikrocontroller generierten Impulse auf. Die Impulse werden über einen Optokoppler in den Stromkreis des Shunts übertragen. Die Bedeutungen der übrigen LEDs sind abhängig von den Akku-Nennspannungen in Tabelle 1 zusammengefasst.

Auch für Bleiakkus mit der Nennspannung 2 V ist der Aktivator geeignet. Die Betriebsspannung 5 V muss dann von einer externen Quelle geliefert werden, oder ein Aufwärts-Spannungswandler von 2 V nach 5 V stellt diese Spannung bereit. Den Status eines 2-V-Akkus können die rote, orange oder grüne LED nicht anzeigen.

Tabelle 1. Die LED-Farben signalisieren die Ladezustände.

Messbereich	**0...10 V**			**0...20 V**	**0...30 V**
Spannung	V < 2,5	2,5 < V < 5	5 < V < 10		
LED	Akku 2 V	Akku 4 V	Akku 6 V	Akku 12 V	Akku 24 V
Rot	< 1,98 V	2,5...4,00 V	5...5,95 V	< 11,9 V	< 23,8 V
Orange	1,98...2,08 V	4,00...4,16 V	5,95...6,25 V	11,9...12,5 V	23,8...25,0 V
Grün	2,08...2,5 V	4,16...5 V	6,25...10 V	> 12,5 V	> 25,0 V

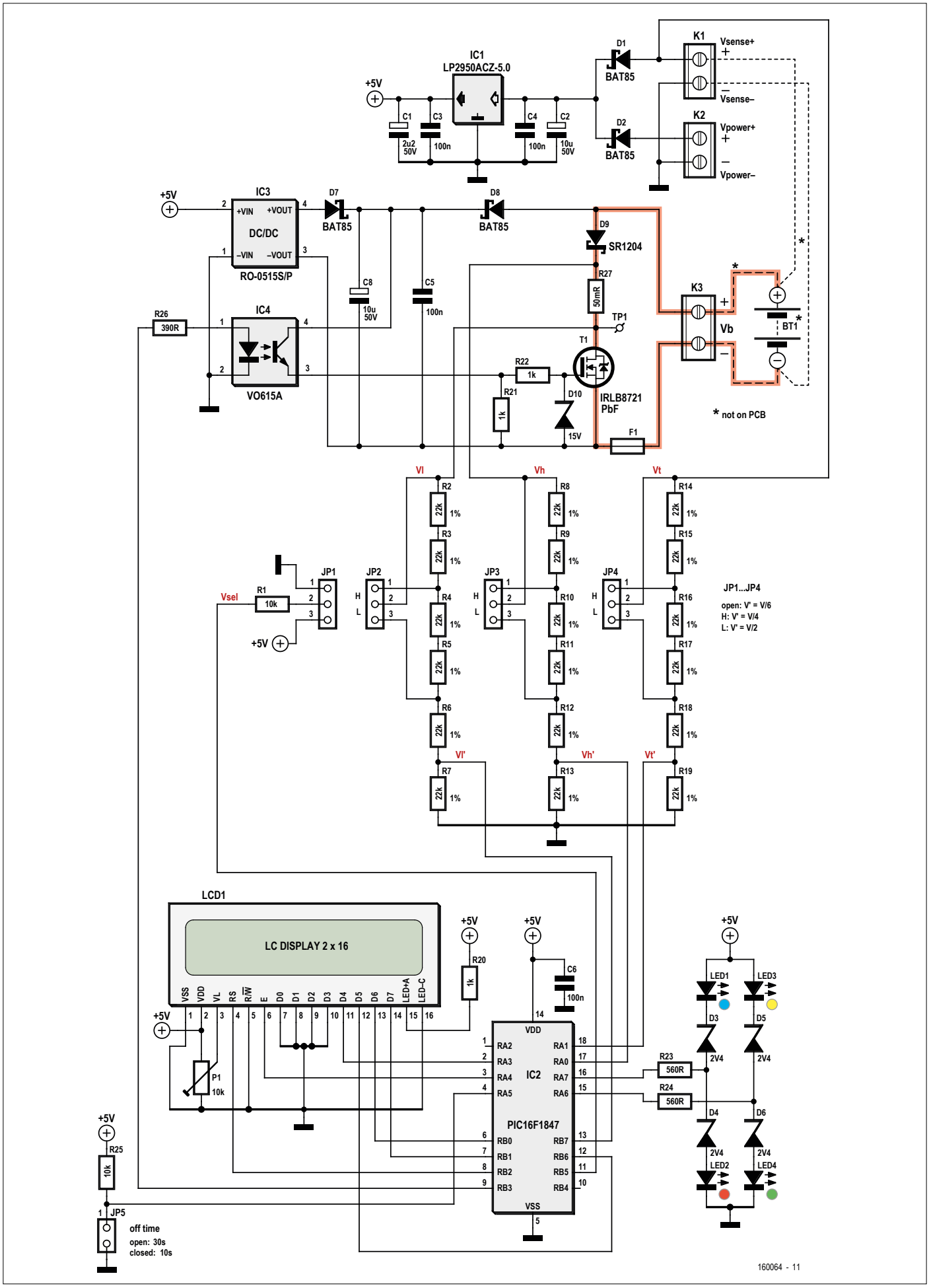

IC1
LP2950ACZ-5.0
+5V
C1
2u2
50V
C3
100n
C4
100n
C2
10u
50V
D1
BAT85
D2
BAT85
K1
Vsense+
Vsense–
K2
Vpower+
Vpower–
IC3
+VIN
+VOUT
DC/DC
–VIN
–VOUT
RO-0515S/P
D7
BAT85
D8
BAT85
C8
10u
50V
C5
100n
D9
SR1204
R27
50mR
TP1
T1
IRLB8721
PbF
K3
Vb
BT1
* not on PCB
R26
390R
IC4
VO615A
R22
1k
R21
1k
D10
15V
F1
Vl
Vh
Vt
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
22k
1%
JP1
JP2
JP3
JP4
Vsel
R1
10k
H
L
JP1...JP4
open: V' = V/6
H: V' = V/4
L: V' = V/2
Vl'
Vh'
Vt'
LCD1
LC DISPLAY 2 x 16
VSS
VDD
VL
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+A
LED–C
P1
10k
R20
1k
C6
100n
IC2
PIC16F1847
RA2
RA3
RA4
RA5
RA1
RA0
RA7
RA6
RB0
RB1
RB2
RB3
RB7
RB6
RB5
RB4
R23
560R
R24
560R
LED1
LED2
LED3
LED4
D3
D4
D5
D6
2V4
R25
10k
JP5
off time
open: 30s
closed: 10s
160064 - 11

Tabelle 2. Übersicht über Warnungen und Betriebsstopps.			
	LED	**Blinken**	**Aktion**
Wahrscheinlich offener Stromkreis	Grün	Langsam (1 s)	Warten bis Abhilfe geschaffen ist.
Keine Klemmenspannung*	Grün	Schnell (400 ms)	Warten bis Abhilfe geschaffen ist.
Klemmenspannung nicht im Messbereich	Orange	Schnell (400 ms)	Betrieb stoppt. *Akku-Aktivator unverzüglich vom Akku trennen!*
Maximaler Strom überschritten	Rot	Schnell (400 ms)	Betrieb stoppt. Externen Widerstand R_{ext} vorschalten und neu starten.
* Dieser Zustand kann nur auftreten, wenn die Mikrocontroller-Schaltung an einer externen Spannungsquelle unter 5,5 V betrieben wird.			

Akkuzustand messen, 0,2 V bis 30 V

Auch wenn wir hier nur die Akkus nennen, gelten die Ausführungen zum Innenwiderstand ebenso für nicht wiederaufladbare Batterien (Primärelemente).

Mit einem Shunt, der den Wert 50 mΩ hat, liegt der Strom abhängig von der Akkuspannung theoretisch zwischen 40 A und 600 A. In der Praxis addieren sich zum Shunt-Widerstand der Akku-Innenwiderstand, die Kabelwiderstände sowie der Durchlasswiderstand der Schottky-Diode (D9). Dies hat zur Folge, dass real deutlich niedrigere Ströme fließen. Anzustreben sind Impulsströme von etwa 10 A im Spannungsbereich 0,2...10 V und 100 A bei höheren Spannungen.

Durch einen zweiten Widerstand, der mit dem Shunt in Reihe geschaltet wird, lässt sich der maximale Strom bei hohen Spannungen herabsetzen. Eine Alternative sind Shunts von 50 mΩ, 100 mΩ oder 220 mΩ für die Spannungsbereiche 0,2...10 V, 5,5...20 V oder 5,5...30 V. Unter 5,5 V muss die Betriebsspannung 5 V einer anderen Quelle entnommen werden. Die Eingangsspannung des A/D-Wandlers darf 5 V nicht übersteigen. Deshalb müssen in den genannten Bereichen Spannungsteiler die Spannung durch 2, 4 oder 6 teilen. Der Teiler ist mit einem Jumper wählbar.

Unter bestimmten Fehlerbedingungen (siehe Tabelle 2) schaltet der Akku-Aktivator ab, im LC-Display erscheint eine Fehlermeldung. Da das LC-Display optional ist, werden die Betriebszustände auch über LEDs signalisiert. Der gravierendste Fehler ist das Überschreiten der Strom-Obergrenze: Wenn dieser Fehler auftritt, leuchtet die rote LED, und der Shunt wird abgeschaltet. Nach Beheben des Fehlers ist ein Neustart des Aktivators nötig.

Negative Werte sind möglich, wenn ein Fehler beim Anschließen des Akkus vorliegt. Die angezeigten Werte hängen von zufällig vorhandenen Ladungen ab. Beim Start des Akku-Aktivators blinken die rote, orange und grüne LED wechselweise.

Werden mehrere Akkus kurz nacheinander gemessen, ist eine kürzere Zykluszeit vorteilhaft. Deshalb lässt sich die Zykluszeit mit einem Jumper von 30 s auf 10 s verkürzen. Bleiakkus müssen stets mit der Zykluszeit 30 s gemessen werden.

Bild 1. In der Schaltung ist der Hochstromkreis mit dem Akku und dem MOSFET hervorgehoben. Darunter befinden sich der Spannungsteiler für die zu messende Spannung sowie die Steuerung und Anzeige.

Sicherheitsvorkehrungen

Bei Systemen, in denen Ströme von 100 A oder höher fließen, sind wirksame Maßnahmen für die Sicherheit unentbehrlich. Davon ist insbesondere der Stromkreis betroffen, in dem der Shunt liegt. Wir nennen nachfolgend drei Aspekte, die für die Sicherheit besondere Bedeutung haben:

1. Die Leitungen im Shunt-Stromkreis müssen den hohen Stromimpulsen gewachsen sein. Insbesondere müssen die Querschnitte der betroffenen Platinen-Kupferbahnen mit Lötzinn verstärkt werden. Die Schottky-Diode, der Shunt-Widerstand und der MOSFET sind nicht nur über die Kupferbahnen, sondern auch über ihre langen Anschlussdrähte miteinander zu verbinden. Der Raum zwischen den Bahnen und den Drähten wird mit Lötzinn aufgefüllt.
2. Die Kabel zum Akku sind kurz zu halten, sie dürfen nicht aufgerollt werden. Aufgerollte Kabel können induktive Lasten bilden, was die Spitzenlasten unerwünscht in die Höhe treibt. Die Software schaltet den überlasteten Stromkreis zwar ab, das kann jedoch erst nach Messen des Stroms geschehen. Dann ist es oft schon zu spät.
3. Falls der MOSFET wegen eines Fehlers zu lange leitet oder defekt wird, brennt die Sicherung F1 durch. Zum Messen von 1,2-V-Akkus wird für F1 eine 3-A-Sicherung (flink) empfohlen, bei höheren Spannungen darf der Wert höchstens 10 A (flink) betragen.

Bleiakku-Aktivator-Schaltung

Zum Schaltungsteil, in dem hohe Ströme fließen, gehören die Schutzdiode D9, der induktionsarme Shunt-Widerstand R27 und der Leistungs-MOSFET T1. Ströme bis etwa 100 A können mit der Gate-Spannung 5,5 V geschaltet werden, für Ströme bis 220 A sind 9 V bis 10 V nötig. Die propagierten Schaltzeiten liegen bei 10 ns, wegen des Widerstands vor dem Gate sind sie jedoch länger.

Der MOSFET-Typ wurde wegen seiner niedrigen Gate-Ladung Qg = 7,6 nC bei 4,5 V ausgewählt. Die resultierende Kapazität beträgt

$$C = Q / V = 7{,}6 / 4{,}5 \approx 2 \text{ nF}.$$

Mit dem Wert 1 kΩ für Gate-Vorwiderstand R22 ist die RC-Zeit

$$Ton = 103 \cdot 2 \cdot 10\text{-}9 = 2 \text{ µs}.$$

Mit R21 ist die Zeit Toff = 4 µs. Für die 100 µs langen Impulse sind dies akzeptable Schaltzeiten.

Der MOSFET wird in den Leitzustand versetzt, indem das Gate in möglichst kurzer Zeit auf genügend Spannung gebracht wird. Da der Gate-Stromkreis und der Mikrocontroller nicht unmittelbar verbunden sein dürfen, sorgt ein Optokoppler für eine galvanische Trennung. Der Optokoppler stellt auch die nötige Energie zum Steuern des Gates bereit. Für diesen Zweck sind spezielle Optogate-Treiber auf

dem Markt.
Wir haben verschiedene Typen getestet, mit dem Ergebnis, dass nicht ein einziges Exemplar mit kurzen Schaltzeiten einen steilen 100-µs-Impuls erzeugen konnte. Meistens hatte entweder nur die Einschaltzeit oder die Ausschaltzeit einen zufriedenstellenden Wert, niemals jedoch beide zugleich. Die Stromversorgung für den Gate-Impuls muss mindestens 8 V liefern, damit Ströme im Bereich 100 A bis 220 A fließen können. Falls die Akkuspannung darunter liegt, muss diese Spannung aus anderer Quelle bezogen werden. Deshalb befindet sich der DC/DC-Wandler IC3, ein RO-0515S/P, in der Schaltung; er ist für die Gate-Spannung bei Akku-Spannungen unter 8 V zuständig. Liegt sie höher, kann der Akku über Diode D8 für die Gate-Spannung sorgen. Schnelle Schaltzeiten gewährleistet ein DC/DC-Wandler, der im Bereich 5...15 V arbeitet, für die minimale Gate-Spannung 8 V ist dies mehr als ausreichend. Die Gate-Spannung darf 18 V nicht übersteigen, deshalb sind die Zenerdiode D10 (15 V) und der Widerstand R22 (1 kΩ) hinzugefügt. Widerstand R21 bewirkt eine schnelle Entladung am Impuls-Ende.

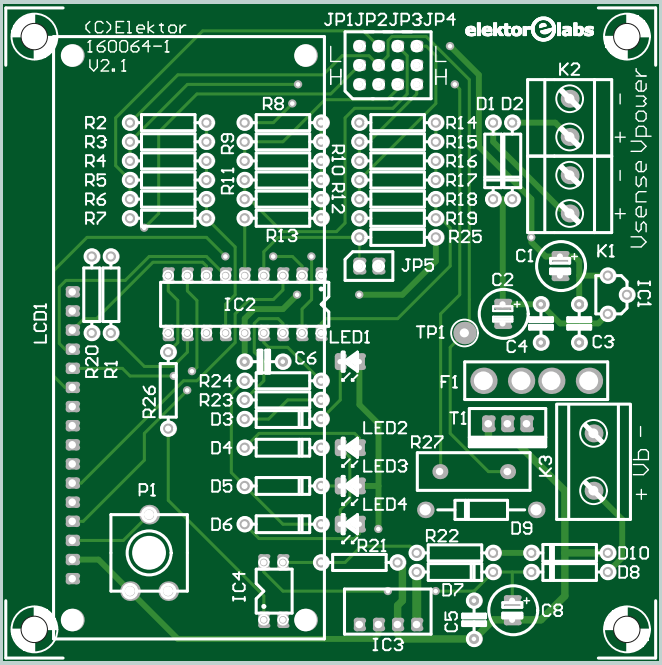
Damit Messungen an Blei-Akkus mit der Nennspannung 6 V ohne externe Spannungsquelle möglich sind, wird ein Low-drop-Spannungsregler verwendet. IC1, ein LP2950C, benötigt beim Ausgangsstrom 30 mA nur 5,3 V an seinem Eingang. Für Messungen unter 5,5 V (zuzüglich 0,2 V an der Schottky-Diode) muss eine externe Spannungsquelle verwendet werden, sie wird an K2 angeschlossen. K2 befindet sich nahe K1, an dem die Klemmenspannung liegt. Die Anschlüsse K1 und K2 dürfen nicht miteinander verwechselt werden! Die Klemmenspannung wird ausschließlich über den Vsense+-Anschluss von K1 gemessen. Eine weitere Warnung gilt den Eingängen dieses Steckverbinders, auch hier kann eine Verwechslung fatale Folgen

STÜCKLISTE

Widerstände: (0,25 W/250 V, wenn nicht anders angegeben)
R1,R25 = 10 k
R2...R19 = 22 k, 1 %, 0,6 W, 350 V
R20...R22 = 1 k
R23,R24 = 560 Ω
R27 = 50 mΩ, 1 W, MPC75
R26 = 390 Ω
P1 = Trimmpoti 10 k

Kondensatoren:
C1 = 2,2 µ, 50 V, RM 2 mm, 5·11 mm
C2,C8 = 10 µ, 50 V, RM 2 mm, 5·11 mm
C3...C6 = 100 n, 50 V, X7R, RM 5,08 mm

Halbleiter:
D1,D2,D7,D8 = BAT85
D3...D6 = BZX79-C2V4
D9 = SR1204
D10 = BZX55C15V
LED1 = LED 3 mm, blau
LED2 = LED 3 mm, rot
LED3 = LED 3 mm, gelb
LED4 = LED 3 mm, grün
T1 = IRLB8721PBF
IC1 = LP2950ACZ-5.0
IC2 = PIC16F1847-I/P, programmiert (EPS 160064-41)
IC3 = RO-0515S/P
IC4 = VO615A



Außerdem:
F1 = Sicherung mit Halter für Platinenmontage (siehe Text)
LCD1 = LC-Display, 2·16 Zeichen, EPS 120061-74
LCD1 = Stiftleiste 16-polig, vertikal, RM 2,54 mm
LCD1 = Buchsenleiste 16-polig, vertikal, RM 2,54 mm
K1,K2 = Kabelschraubklemme 2-polig, RM 5,08 mm
K3 = Kabelschraubklemme 2-polig, RM 7,62 mm
JP1...JP4 = Stiftleiste 3-polig, vertikal, RM 2,54 mm
JP5 = Stiftleiste 2-polig, vertikal, RM 2,54 mm
Jumper für JP5
Buchsenleiste 2x5-polig, vertikal, für JP1...JP5
IC-Fassung DIP-18, für IC2
Platine 160064-1, Version 2.1

haben. An Port A1 des Mikrocontrollers würde negative Spannung liegen, was der Mikrocontroller sicher nicht toleriert.

Die Spannung des Bleiakkus wird von drei LEDs signalisiert (LED2, LED3 und LED4, Rot, Orange und Grün). Die blaue LED (LED1) macht den Stromimpuls sichtbar, für die Sichtbarkeit wird die Einschaltdauer über die Impulsdauer hinaus verlängert. Da jeweils zwei LEDs mit zwei 2,4-V-Zenerdioden in Reihe geschaltet sind (D3 und D4 oder D5 und D6, beide zusammen 4,8 V), leuchtet bei 5 V keine der beiden LEDs auf. Dies gilt für den Fall, dass sich der Mikrocontroller-Ausgang im Tristate-Zustand befindet. Wenn der Ausgang abhängig von der zu aktivierenden LED High oder Low wird, reicht die Spannung für die betreffende LED aus. Die LED-Botschaften werden von weiteren Informationen auf dem optionalen LC-Display unterstützt.

Der Mikrocontroller PIC16F1847 (IC2) ist der am umfangreichsten ausgestattete 8-bit-Typ im 18-Pin-Gehäuse von Microchip. Ausgang B3 steuert über Optokoppler IC4 das Gate des MOSFETs T1. Die unbelastete Klemmenspannung V0 wird unmittelbar vor dem Stromimpuls gemessen, während des 100 µs breiten Impulses folgen die Spannungen Vt (belastete Klemmenspannung), Vh (Shunt hoch) und Vl (Shunt niedrig). Die zugehörigen A/D-Wandler-Eingänge sind AN1, AN0 und AN6.
Beim ersten Start muss die Spannung am MOSFET zwischen TP1 (Drain) und dem Minus-Anschluss von K3 (Source) gemessen werden. Dabei darf die Spannung nicht merklich absinken.

Die Mikrocontroller-Schaltung (Mikrocontroller, LEDs, LC-Display und der Spannungsregler 5 V) hat einen Strombedarf von ungefähr 7 mA, der DC/DC-Wandler nimmt 16...19 mA auf. Das ist zwar nicht unbedingt sparsam, doch bei kurzen Messungen des Innenwiderstands von Zellen 1,2...1,5 V dürfte eine 9-V-Batterie einige Zeit durchhalten.

Spannungsmessungen und Jumper

Einige Einstellungen der Software werden mit den Jumpern JP2, JP3 und JP4 vorgenommen. JP1 gehört zu einem Tristate-Eingang, er kann Low, High oder offen sein. Der 10-kΩ-Widerstand schützt die Portleitung, da der Eingang zum Abfragen des Status auch kurzzeitig als Ausgang geschaltet ist.
Die zu messenden Spannungen Vt, Vh und Vl werden an die maximale Spannung 5 V der A/D-Wandler-Eingänge mit den Spanungsteilern R2...R19 angepasst. Spannungen bis 30 V werden mit dem Faktor 6 herabgeteilt. An der Spannung liegen sechs in Reihe geschaltete, identische Widerstände, so dass die herabgeteilte Spannung am letzten Widerstand abgegriffen werden kann. Sollen Spannungen im Bereich 0...20 V gemessen werden, überbrückt Jumper JP2 in Position H die Widerstände R2 und R3, der Teilfaktor beträgt dann 4. Mit dem Jumper in Position L werden R2...R5 überbrückt, um den Teilfaktor auf 2 für den Bereich 0...10 V herabzusetzen. Jumper JP1 dient dazu, dem Mikrocontroller zu signalisieren, welcher Teilfaktor aktiv ist. Die vier Jumper JP1...JP4 liegen nebeneinander auf einer gemeinsamen Stiftkontaktleiste. Wie Bild 2 zeigt, wurde eine Buchsenleiste mit 2 · 4 Kontakten zu einem vierfachen Jumper verschaltet.

Die Zykluszeit der Stromimpulse lässt sich mit JP5 auf 30 s (kein Jumper) oder 10 s (Jumper aufgesteckt) einstellen. Bleiakkus erfordern stets die Zykluszeit 30 s.
Innerhalb der Impulsdauer 100 µs müssen drei Spannungsmessungen vorgenommen werden. Ob diese Zeit ausreicht, hängt sowohl von der Hardware als auch von der Software ab. Die zeitkritischen Komponenten sind der Optokoppler, der MOSFET und der Mikrocontroller. Beim Optokoppler VO615A betragen die Anstiegszeit (Ton) und die Abfallzeit (Toff) mit der Last 1 kΩ etwa 9...10 µs. Für drei A/D-Wandler-Messungen stehen folglich ungefähr 100-(2 · 10) = 80 µs zur Verfügung.

Beträgt die Taktfrequenz des Mikrocontrollers 16 MHz, vergeht mindestens 1 µs, um ein Bit umzusetzen (Tad). Ein 10-bit-A/D-Wandler benötigt insgesamt 11,5 Tad, was 11,5 µs entspricht. Die Akquisitionszeit Tacq setzt sich zusammen aus der Verstärker-Einstellzeit, der Ladezeit des Hold-Kondensators TC und einer temperaturabhängigen Verzögerung: 2,0 µs + TC + 0,05 ΔT (bei 25 °C). Die Zeit TC hängt unter anderem vom Quellenwiderstand Rs ab, sie beträgt beim verwendeten Mikrocontroller

7,62 · 10 pF · (8 kΩ + Rs).

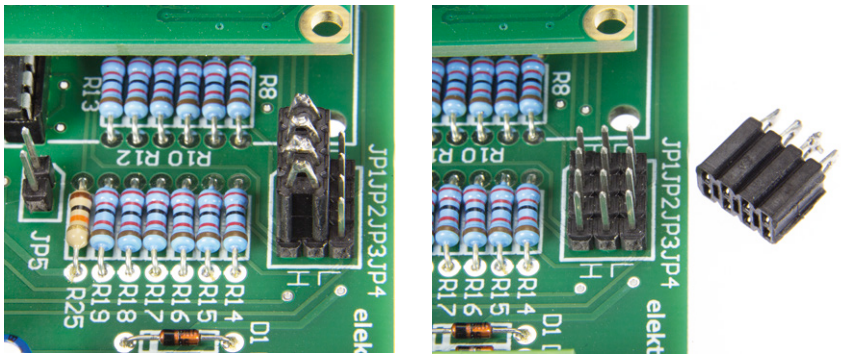


Bild 2. Für das Festlegen des Spannungsbereichs wurde aus einer Buchsenleiste (2 · 4 Kontakte) ein Vierfach-Jumper konstruiert.

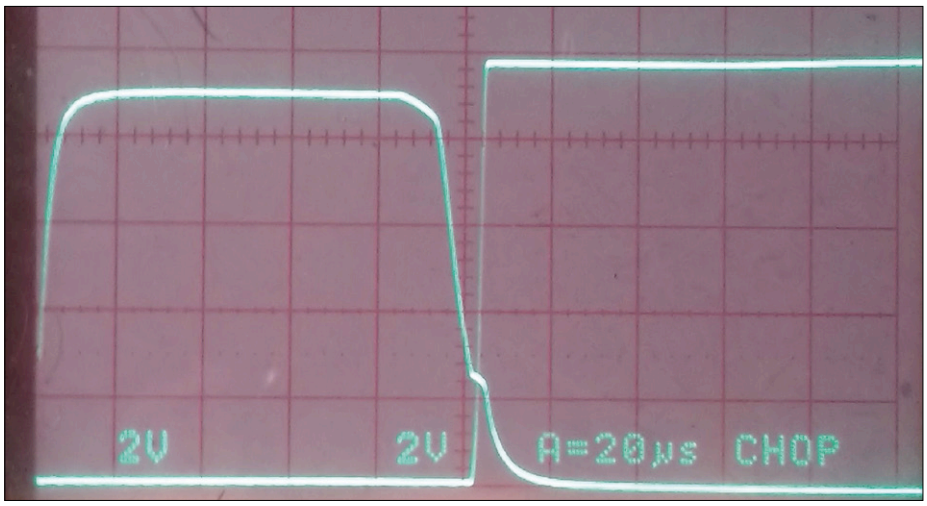


Bild 3. Kurve links: Gate-Spannung Vgs mit der Impulszeit 94 µs bei drei A/D-Wandler-Messungen. Kurve rechts: Spannungsverlauf am MOSFET (Vds), die gesamte Shunt-Zeit lag bei ungefähr 105 µs (Vgs > 3 V beim Shunt-Strom 0,4 A und Vo = 10 V).

Für den optimalen Quellenwiderstand 10 kΩ ist dann TC = 1,4 µs. Im vorliegenden Fall gilt Rs = 22 kΩ || 22 kΩ = 11 kΩ beziehungsweise Rs = 22 kΩ || (5 · 22 kΩ) = 18 kΩ, so dass sich für TC = 1...2 µs ergeben. Die Abhängigkeit von der Temperatur ist vernachlässigbar. Die gesamte A/D-Wandler-Akquisitionszeit (Tacq) beläuft sich dann bei 25 °C auf ungefähr 11,5 + 2 + 2 = 15,5 µs. In der Software wird für TC die längere Zeit 10 µs angenommen, so dass Tacq = 24 µs ist. Drei Messungen des A/D-Wandlers dauern folglich 72 µs, eine Zeit, die mit einigem Abstand innerhalb der verfügbaren 80 µs liegt.

Für jeden Programmschritt benötigt die Software die Zeit 0,25 µs (ADC-Enable, Read, Save und Disable). Mit den standardmäßigen Prozeduren von Flowcode 6 dauern die drei A/D-Wandler-Messungen ungefähr 150 µs, also zu lange. Durch Ersetzen der Flowcode-Prozeduren durch Prozeduren in C konnten die Zeiten um etwa ein Drittel verkürzt werden, so dass die drei Messungen innerhalb von 80 µs ausgeführt werden können.

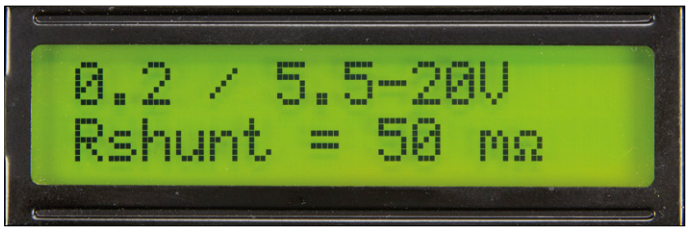
Wie Bild 3 beweist, stimmen die für das Gate berechneten Schaltzeiten mit der Praxis gut überein. Dort ist auch erkennbar, dass Vds deutlich steiler verläuft als Vgs. Die Anstiegszeit von Vds beträgt ungefähr 0,8 µs, während die Abfallzeit bei 3 µs liegt. Dank dieser kurzen Zeiten muss der MOSFET nicht gekühlt werden.

Software in Flowcode

Die Software wurde in Flowcode 6 geschrieben, die Flowcode-Datei und die zugehörige Hex-Datei können von der Projektseite [4] heruntergeladen werden. Nach dem Start werden der Faktor des Spannungsteilers, der Spannungsbereich und die Zykluszeit angezeigt. Das Programm wartet, bis eine Spannung am Shunt-Widerstand gemessen werden kann. Anschließend zeigt das LC-Display in der ersten Zeile die unbelastete Klemmenspannung V0 und die belastete Klemmenspannung Vt an. In der zweiten Zeile erscheinen der Impulsstrom in A und der Akku-Innenwiderstand in mΩ. Außerdem zählt ein Countdown-Zähler die Sekunden bis zum nächsten Stromimpuls. Die Anzeigegenauigkeit der vier Werte beträgt mindestens drei Digits. Die tatsächliche Genauigkeit hängt von mehreren Faktoren ab, sie genügt, um vom Innenwiderstand des Akkus auf den Betriebszustand schließen zu können.
Nach Anschließen des Akkus wartet das Programm eine Sekunde, um Spannungssprünge zeitlich zu überbrücken. Der Startvorgang wird durch das wechselweise Blinken der roten, orangen und grünen LED begleitet.
Wenn die Flowcode-Software modifiziert und der Controller neu programmiert werden soll, muss das Taktsignal auf „Intern“ und 16 MHz gesetzt werden, der externe MCLR und die Low-voltage-Programmierung müssen deaktiviert werden (Configuration words: config1 0x09A4 und config2 0x1CFF).

Ergebnisse in der Praxis

Der Akku-Aktivator wurde zuerst am Bleiakku eines Autos erprobt. Da das Fahrzeug regelmäßig nur Kurzstrecken gefahren wird, hatte der Innenwiderstand mit 6 mΩ einen relativ hohen Wert. Das Fahrzeug ließ sich aber stets problemlos starten.
Das Diagramm in Bild 4 demonstriert den wiederherstellenden Effekt der Stromim-



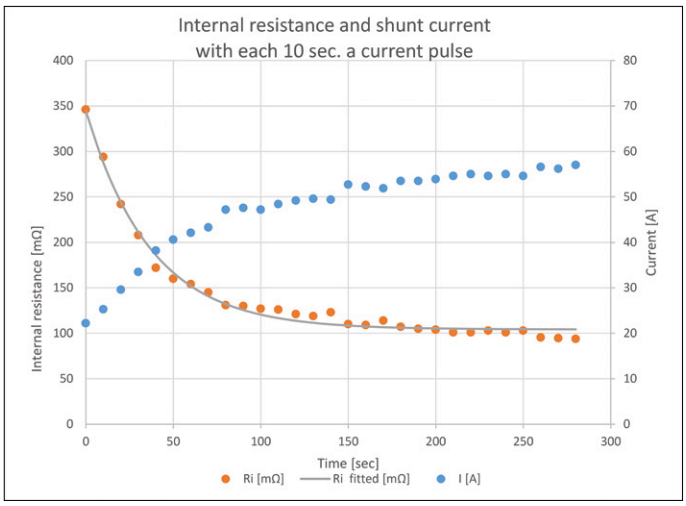


Bild 4. Der Regenerierungseffekt bei einem schwächelnden Bleiakku, die Spannung an den Klemmen lag bei 10,3 V.

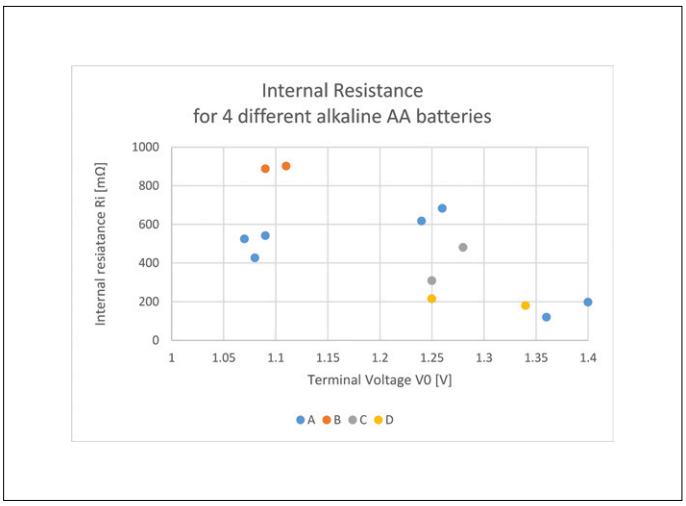


Bild 5. Die Innenwiderstände von Alkaline-Batterien des Typs AA (LR6) als Funktion der unbelasteten Klemmenspannung. Gemessen wurden vier Markenprodukte, bezeichnet mit A, B, C und D.

pulse bei einem anderen Akku. Schon nach vier bis fünf Impulsen hat sich der Innenwiderstand halbiert, er nähert sich schnell dem Endwert.

Bei einem 12-V-Bleiakku, Kapazität 7 Ah, wurde eine Temperaturabhängigkeit des Innenwiderstands von annähernd −0,7 mΩ/°C gemessen. Der Innenwiderstand stieg von 34 mΩ bei Zimmertemperatur auf 62 mΩ bei −18 °C in einem Tiefkühlschrank.

Ein neuer Bleiakku (12 V, 7 Ah) hatte nach dem Laden den Innenwiderstand 34 mΩ ±2 %, ein identischer Akku wies nach acht Jahren den Innenwiderstand 52 mΩ ±3 % auf. Die Klemmenspannungen betrugen bei beiden 13,2 V.
An zwölf NiMH-Akkus, Kapazität 4600 mAh, verbunden über Lötfahnen mit niedrigen Kontaktwiderständen, wurde der Innenwiderstand 101 mΩ gemessen. Nach dem Laden mit 2000 mAh war der Innenwiderstand auf 86 mΩ gesunken.

Bei Bleiakkus lag die Messgenauigkeit bei 1 % oder sogar darunter. Bei AA- und AAA-Zellen (Alkaline und NiMH) traten dagegen Toleranzen von 10 % oder mehr auf. Das bedeutet, dass eine einzelne Messung nicht ausreicht. Das Ergebnis ist nur verlässlich, wenn der Mittelwert aus mehreren Messungen gebildet wird. Schließlich haben wir noch Alkaline-Batterien unterschiedlicher Marken gemessen, das Ergebnis geht aus Bild 5 hervor. Hier fällt auf, dass die Innenwiderstände abhängig von der Marke stark streuen. Ein sehr bekanntes Produkt schneidet am schlechtesten ab, oder ist das nur Zufall? Dass der Innenwiderstand mit zunehmender Entladung steigt, lässt sich deutlich erkennen. Um die Unterschiede unter den Marken zu erklären, bedarf es einer umfassenden Analyse, die unser Bleiakku-Aktivator leider nicht leisten kann. ⏮

(160064)gd

EINKAUFSZETTEL

→ **160064-1**
Platine

→ **160064-41**
Programmierter Controller

→ **160064-71**
Kit

→ **120061-74**
LC-Display 2·16 Zeichen

Weblinks

[1] Conrad Batteriewächter, Artikelnummer 191123: www.conrad.de/de/batteriewaechter-baustein-h-tronic-12-vdc-191123.html

[2] A lead acid battery desulfatation tutorial: www.chargingchargers.com/tutorials/battery-desulfation.html

[3] Battery and Energy Technologies in Electropaedia: www.mpoweruk.com/performance.htm

[4] Software-Download: www.elektormagazine.de/160064

Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

Andonstar USB-Mikroskop V160

Bei der Auswahl eines preisgünstigen USB-Mikroskops hängt die Brauchbarkeit des kleinen Geräts nicht so sehr von der Anzahl der Megapixel oder dem Vergrößerungsfaktor ab, es kommt auf die mechanischen Eigenschaften an, die entscheiden, ob das Mikroskop einen festen Platz auf dem Labortisch erobern kann oder schnell in der „Müll-Schublade" verschwindet. Die meisten USB-Oszilloskope werden leider mit einem mangelhaften, instabilen Stativ geliefert, bei dem die geringste Erschütterung des Labortisches oder ein leichtes Berühren der Kamera ausreichend ist, damit die Fokussierung verloren geht. Preisgünstig vielleicht, aber sicher nicht preiswert!

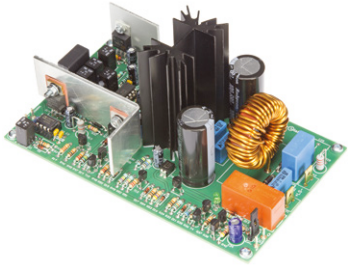
Der Andonstar V160 ist in dieser Beziehung eine löbliche Ausnahme. Dieses Mikroskop besitzt ein kräftiges Stativ, das ein exaktes und stabiles Bild garantiert. Und es ist alles präzise einzustellen.

Luc Lemmens (Elektor-Labor)

www.elektor.de/andonstar-v160

Elektor-Bestseller

1. D-Watt
   www.elektor.de/d-watt



2. ESP8266-WLAN-Funkmodul
   www.elektor.de/150445-91
3. BBC micro:bit
   www.elektor.de/bbc-micro-bit
4. Das ESP8266-Praxisbuch
   www.elektor.de/esp8266-buch
5. Raspberry Pi 3
   www.elektor.de/rpi-3
6. Arduino-Sanduhr
   www.elektor.de/sanduhr
7. Swiss Pi
   www.elektor.de/swiss-pi
8. Elektor Uno R4
   www.elektor.de/elektor-uno-r4

Projekte mit dem RFID-Starterkit für Arduino

Der Arduino ist inzwischen zu einer festen Größe in der Maker-Welt geworden. Der Einstieg in die Controller-Technik ist damit nicht mehr nur Experten vorbehalten. Anders sieht es aus, wenn es um Hardware-Erweiterungen geht. Hier ist der Anwender immer noch weitgehend auf sich selbst gestellt. Wenn man wirklich innovative Projekte umsetzen möchte, muss man sich direkt mit elektronischen Bauelementen befassen.

Mitgliederpreis: 29,80 €

www.elektor.de/projekte-rfid-kit-arduino

Das ESP8266-Praxisbuch

Der ESP8266 ist ein programmierbares WLAN-Funkmodul mit zahlreichen Schnittstellen wie UART, I²C und SPI. Das Board ist sehr preiswert und bereits für unter 3 Euro verfügbar. Die UART-Schnittstelle sorgt dabei für eine einfache Integration in Mikrocontrollerprojekte. Das ESP8266-Modul kann hervorragend mit dem Arduino zusammenarbeiten und ermöglicht ihm über die serielle Schnittstelle den Zugang zum Netzwerk und Internet.

Mitgliederpreis: 39,80 €

www.elektor.de/esp8266-buch

Digitale Systeme mit FPGAs entwickeln

FPGAs sind Standard-ICs, die dem Anwender die Konfiguration von Hardwarestrukturen ermöglichen. Mit FPGAs ist die Umsetzung digitaler Systeme ab Stückzahl eins machbar. Durch die Verfügbarkeit kostenloser Lizenzen und preiswerter Entwicklungsboards ist die finanzielle Einstiegsschwelle in diese Technik niedrig. Bei der Überwindung der fachlichen Schwelle hilft dieser 6-teilige Kurs.

Mitgliederpreis: 49,00 €

www.elektor.de/digitale-systeme-fpga

Hexadoku

Sudoku für Elektroniker

Wie in jeder Ausgabe finden Sie auch in diesem Heft unser ganz spezielles Sudoku. PC, Oszilloskop und Lötkolben können sich erholen, während Ihre kleinen grauen Zellen auf Hochtouren arbeiten. Wenn Sie alle Hex-Ziffern in den grauen Kästchen herausgefunden haben, sollten Sie uns diese gleich zumailen – denn hier warten drei Elektor-Gutscheine!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt. Wer das Rätsel löst – sprich die Zahlen in den grauen Kästchen herausfindet – kann einen von fünf Gutscheinen im Wert von 50 Euro gewinnen!

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 23. März 2017.

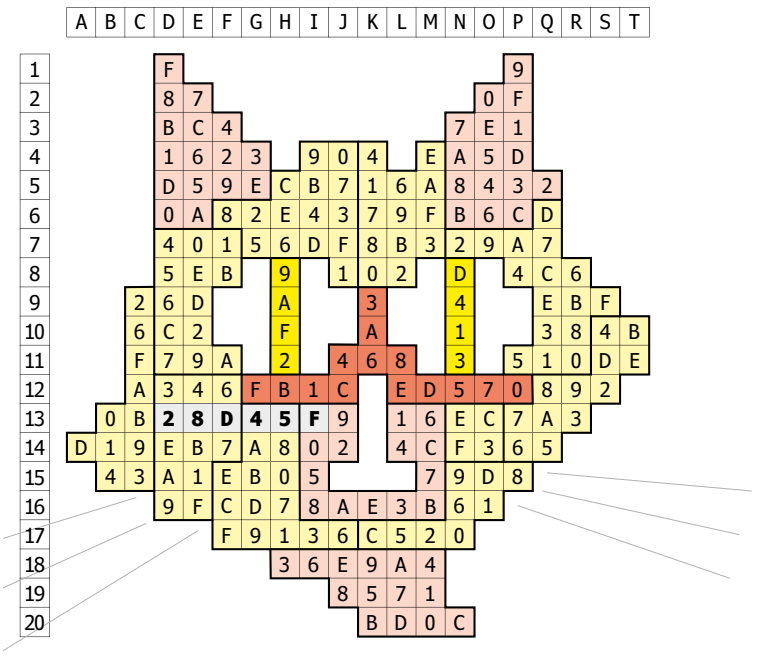
Die Gewinner des Hexadokus aus der Januar/Februar-Ausgabe stehen fest!

Die richtige Lösung ist: **28D45F.**

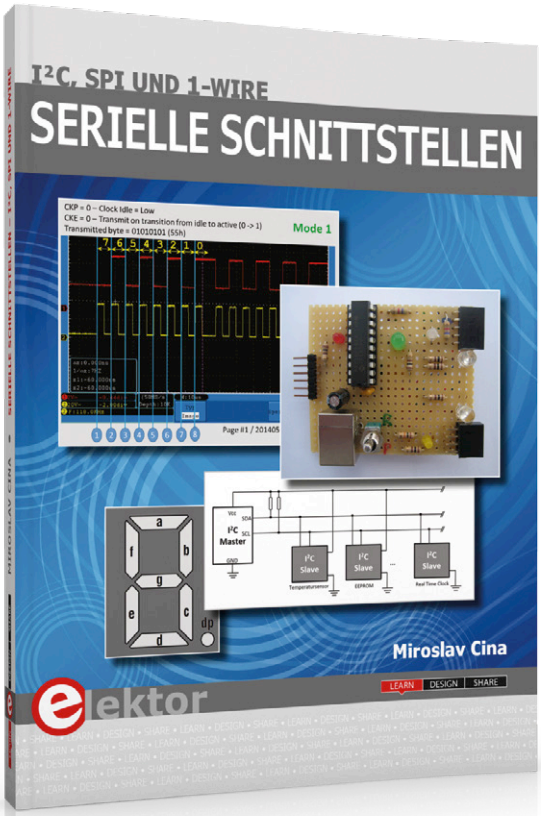
Einen Elektor-Wertgutschein über je 50 € haben gewonnen: Stefan Falcke, Jean-Claude Carré und Harald Metzen.

Die Gewinner des Hexadokus Dezember 2016 finden Sie unter www.elektormagazine.de/hexadoku201612.

		8				4	A	D	C				3		
	A	7											9	4	
C		1		B	9					8	A		5		0
F		3			E	0	2	9	5	7			1		8
				9	A					4	F				
	D		A			8	7	1	6			2		B	
2		4	3	C		B			8		E	9	A		D
	6		8									5		F	
3	0	C				E			9				8	5	1
					1		0	3		F					
	8		1				5	B				A		9	
	B	2		6		A			1		C		D	0	
	C	F	B	7	D					9	0	8	6	E	
8					B	5			D	6					9
9				A	2					1	8				4
D			E	4			9	C			5	7			A



Der Rechtsweg ist ausgeschlossen. Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.



Serielle Schnittstellen

NEU!

Die serielle Schnittstelle dient dem Datenaustausch zwischen Mikrocontrollern und Peripheriegeräten. Bei einer seriellen Datenübertragung werden die Bits nacheinander – also seriell – über eine Leitung übertragen. Das Wissen um diese seriellen Schnittstellen erweitert die Einsatzmöglichkeiten von Mikrocontrollern und Peripheriegeräten ungemein.

In diesem Buch werden die Grundlagen der Kommunikation für die drei seriellen Schnittstellen I²C, SPI und 1-Wire vorgestellt. Anhand praxisnaher Beispiele werden die Schnittstellen anschaulich dargestellt und deren Einsatz verdeutlicht. Software- und Hardwarebeispiele, die man in diesem Buch finden kann, wurden in Assembler für PIC-Mikrocontroller von Microchip realisiert. Die verwendeten Algorithmen und Vorgehensweisen kann man aber auf beliebige Plattform übertragen.

Mitgliederpreis: 33,00 € (frei Haus)
www.elektor.de/serielle-schnittstellen

PORSCHE 6-Zylinder-Boxermotor

Der Porsche-Boxermotor ist einzigartig und die Fahrzeuge, die er antreibt, sind legendär. Mit diesem Paket bauen Sie in rund drei Stunden ein transparentes Funktionsmodell des 2-Liter-6-Zylinder-Boxermotors aus dem Jahr 1966. Erleben Sie mit Ihrer Familie Technikfaszination vom Feinsten!

Mitgliederpreis: 152,10 €

www.elektor.de/porsche-flat-six-boxer-engine

AVR-Programmierung für Quereinsteiger

Atmel AVR ist eine 8-bit-Mikrocontroller-Familie des Herstellers Atmel. Diese Controller sind wegen ihres einfachen Aufbaus, ihrer leichten Programmierbarkeit, den kostenlosen Entwicklungswerkzeugen und der Verfügbarkeit in DIP-Gehäuseformen auch bei Makern äußerst beliebt. Darüber hinaus sind diese Controller bereits ab zwei Euro erhältlich. Im Arduino Uno-Board wird der ATmega328 verwendet.

Mitgliederpreis: 34,80 €

www.elektor.de/avr-prog-fuer-quereinsteiger

Arduino-Sanduhr

Auf der Basis eines Arduino Uno dient dieses coole Gadget zur Anzeige der aktuellen Uhrzeit, indem diese in eine Sandschicht „graviert" wird. Nach einer einstellbaren Zeit „löschen" zwei Vibrationsmotoren die Anzeige, bevor die neue Zeit geschrieben wird. Die Aktivitäten der Sanduhr werden durch zwei Servo-Motoren vorgenommen, die einen Pantograf-Mechanismus antreiben.

Mitgliederpreis: 107,96 €

www.elektor.de/sanduhr

LEARN DESIGN SHARE

Das ESP8266-Praxisbuch

Mit NodeMCU und ESPlorer

Der ESP8266 ist ein programmierbares WLAN-Funkmodul mit zahlreichen Schnittstellen wie UART, I²C und SPI. Das Board ist sehr preiswert und bereits für unter 3 Euro verfügbar. Der Bestseller-Autor Erik Bartmann hat sich ausführlich mit dem ESP8266 und dem NodeMCU beschäftigt. Heraus gekommen ist dabei Das ESP8266-Praxisbuch, in dem er die Leser Schritt für Schritt in die Arbeit mit diesen neuen, preiswerten Bauteilen einführt, mögliche technische Stolpersteine ausführlich behandelt und in zahlreichen Projekten die Praxistauglichkeit – angefangen bei einem selbst gebauten Webserver bis hin zu klugen Relay-Ansteuerung – belegt.

➡ **www.elektor.de/esp8266-buch**

Sensoren am Arduino

Mit Sensoren den Arduino zum Leben erwecken

Mit einem Arduino-Board ohne zusätzliche Sensoren kann man nicht viel anfangen. Dieses Buch richtet sich an jeden, der seinem Arduino-Uno-Board mit Hilfe von zahlreichen Sensoren Leben einhauchen möchte. Wie das geht, zeigt der Autor Schritt für Schritt mit zahlreichen Abbildungen, und das in einer leicht verständlichen Sprache. Als Einstieg in die faszinierende Welt der Sensorik wird in diesem Buch auf das auch bei Elektor erhältliche 37 Module umfassende Sensor-Kit zurückgegriffen.

➡ **www.elektor.de/sensoren-am-arduino**

LabVIEW 3

Für den Praktiker

Nachdem in den Bänden LabVIEW 1: Einstieg in die Praxis und LabVIEW 2: Arrays und serielle Daten zur Einführung in LabVIEW ein breit gefächertes und solides Grundlagenwissen vermittelt wurde, stehen mit LabVIEW 3: Für den Praktiker fortgeschrittene Programmiertechniken auf dem Programm. Die Entwicklung eigener Sub-VIs (Unterprogramme), das Arbeiten mit Schieberegistern und Zustandsautomaten und die Ereignisgesteuerte Programmierung stellen die Schwerpunkte des dritten Bandes dar.

➡ **www.elektor.de/labview-3**

elektor

reichelt elektronik

Technik verbindet.

Jetzt Newsletter abonnieren!

Unter allen neuen Newsletter-Abonnenten wird monatlich ein **technisches Highlight** verlost!

http://**rch.lt/v3**

reichelt.de | @reichelt_el | google.com/+reichelt

„Das rechnet sich ganz schnell"

V-TAC® INNOVATIVE LED LIGHTING

PREISHIT

VT-7126

0,69

A+

LED-LEUCHTMITTEL, GU 10

- Leistung: 3 Watt
- Lichtleistung: 210 lm
- Lichtfarbe: warmweiß
- Lebensdauer: 20.000 h
- Abstrahlwinkel: 110°
- CRI (Ra): 80

JETZT BELEUCHTUNG ERNEUERN UND SPAREN!

LED-STREIFEN

- Leistung 18 Watt
- Abstrahlwinkel: 120°
- Länge: 5 m
- Breite: 8 mm
- flexible und einfache Installation
- CRI (Ra): 80

A

je 5,35

VT-2016	5,35	warmweiß
VT-2041	5,35	neutralweiß
VT-2005	5,35	kaltweiß

LED-LEUCHTMITTEL, E 27

Mit diesen LED-Leuchtmitteln lassen sich nahezu alle Energiesparlampen und herkömmliche Lampen im E27-Sockel ersetzen. So können Sie sofort mit dem Stromsparen beginnen.

- Leistung: 10 Watt
- Lichtleistung: 806 lm
- Lichtfarbe: warmweiß
- Lebensdauer: 20.000 h
- CRI (Ra): 80

A+

VT-4209

1,69

LED-STREIFEN

- Leistung 18 Watt
- Abstrahlwinkel: 120°
- Länge: 5 m
- Breite: 8 mm
- flexible und einfache Installation

VT-2015	5,35	rot
VT-2011	5,35	grün
VT-2013	5,35	blau
VT-2124	7,95	RGB

A

ab 5,35

MEHR LED-LICHTTECHNIK

Die Artikel dieser Anzeige und viele weitere finden Sie in unserem Online-Shop: http://**rch.lt/zW**

Tagespreise · Preisstand: 31. 1. 2017, Preise in € inkl. gesetzl. MwSt., zzgl. Versandkosten, reichelt elektronik, Elektronikring 1, 26452 Sande (D)

- über 45 Jahre Erfahrung
- schneller 24-Std.-Versand
- über 70.000 Produkte am Lager
- kein Mindermengenzuschlag
- zertifiziert nach DIN EN ISO 9001:2008

www.reichelt.de

BESTELLHOTLINE: +49 (0)4422 955-333